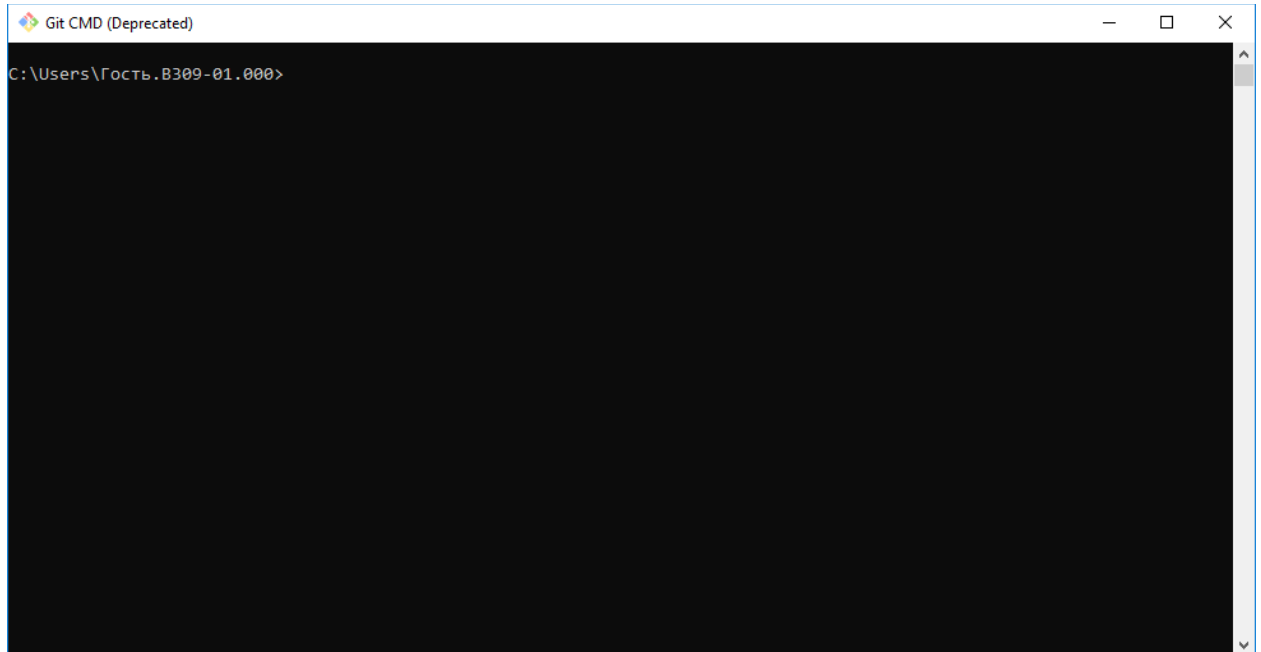


Руководство по использованию github + git

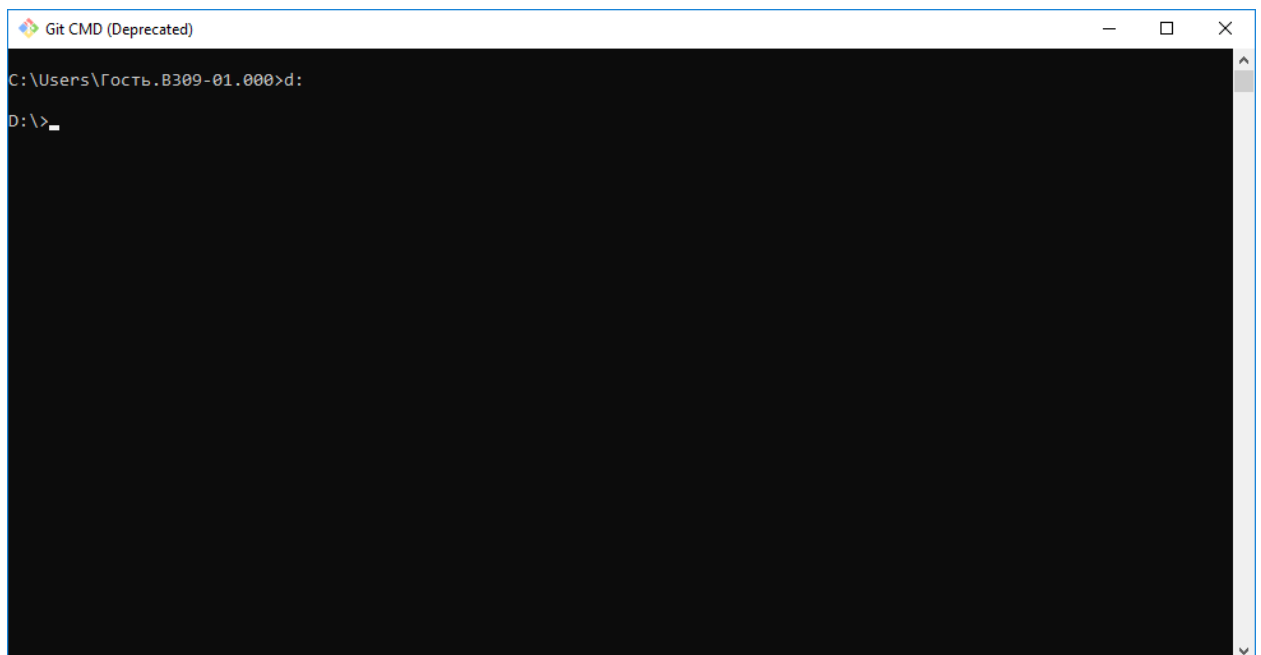
Для начала необходимо зайти на сайт github.com и зарегистрироваться.

Для работы с git необходимо открыть меню Пуск и ввести Git CMD, открываем его.

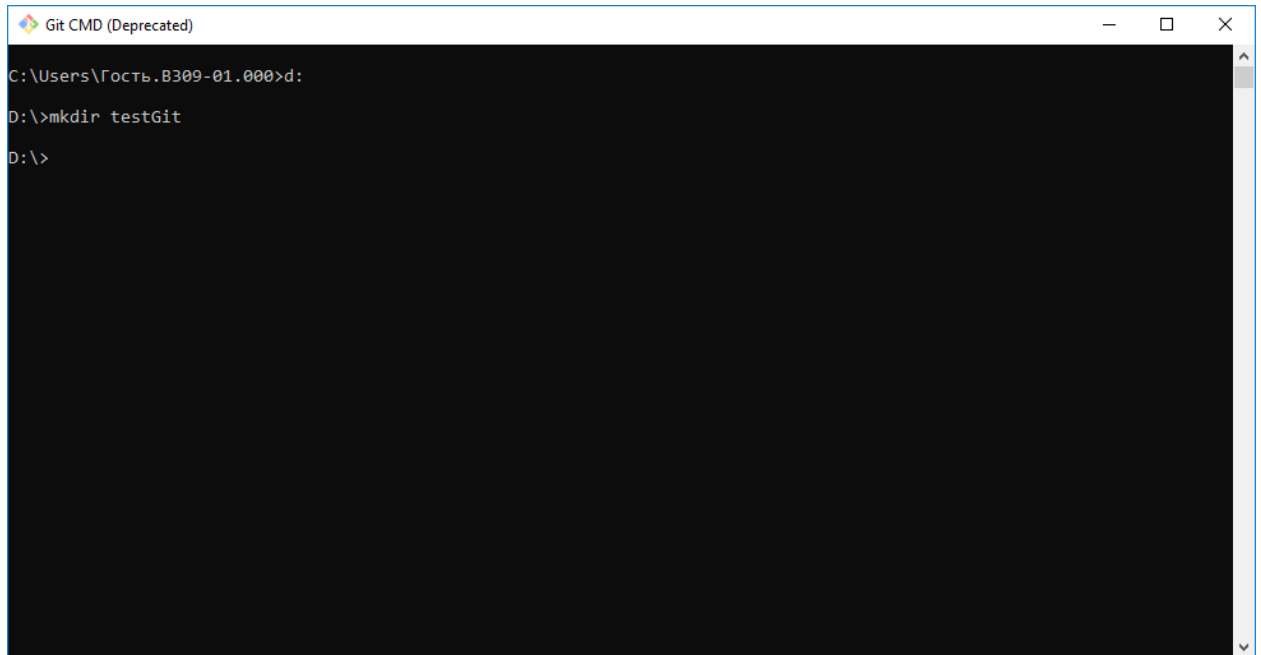
Так выглядит его интерфейс:



Далее нам необходимо перейти в нужную нам папку где мы будем создавать репозиторий. Переход осуществляется путем ввода в git команду **cd путь:**, для перехода на уровень ниже, необходимо написать **cd ..** для перехода на другой диск можно просто написать **d:**

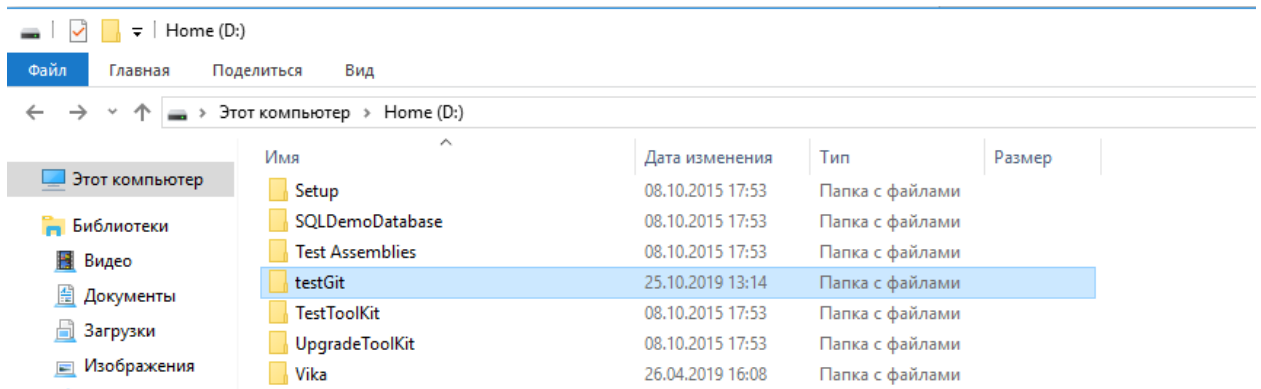


Если же вам необходимо создать папку в которой будет находиться репозиторий, тогда необходимо прописать: **mkdir название_папки**, в моем случае я создам папку с названием testGit

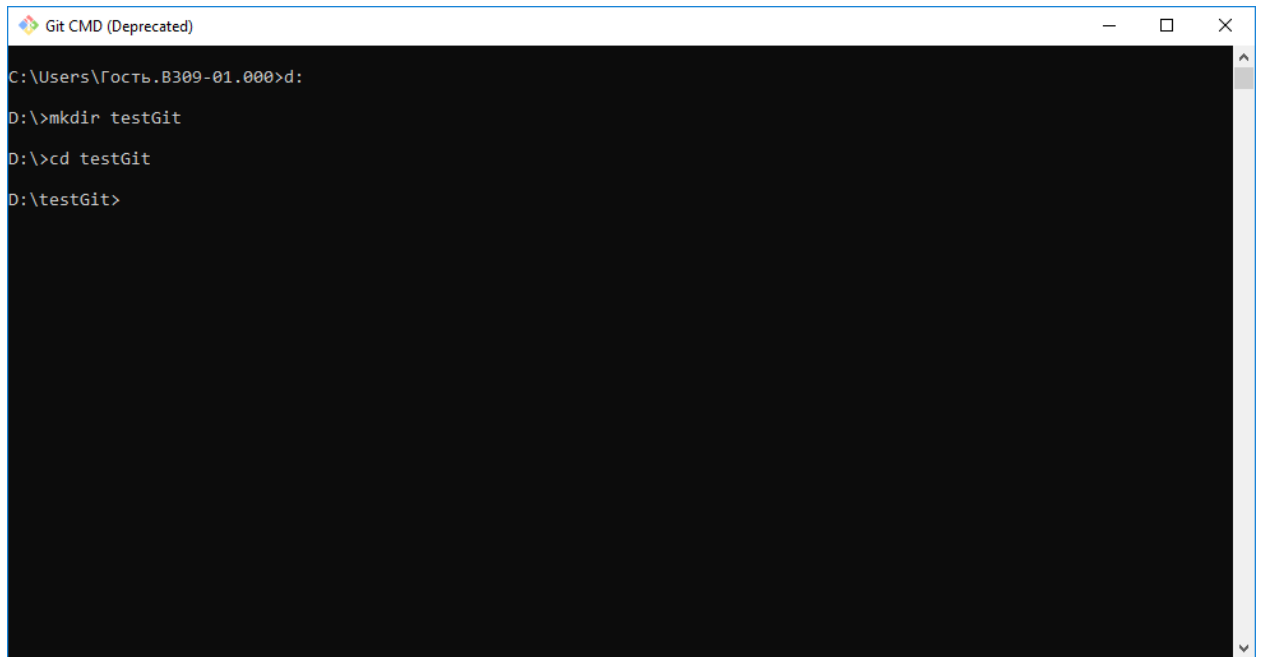


```
Git CMD (Deprecated)
C:\Users\Гость.В309-01.000>d:
D:\>mkdir testGit
D:\>
```

После этого на диске D: появится папка.



Командой `cd` название_папки мы переходим непосредственно в нашу папку

A screenshot of a terminal window titled "Git CMD (Deprecated)". The terminal shows the following commands and their outputs:

```
C:\Users\Гость.В309-01.000>d:
D:\>mkdir testGit
D:\>cd testGit
D:\testGit>
```

Если вам необходимо создать репозиторий уже в имеющейся папке, то вам необходимо просто в нее перейти, создавать ничего не нужно.

Репозиторий – это место, где хранится и поддерживается какие-либо данные.

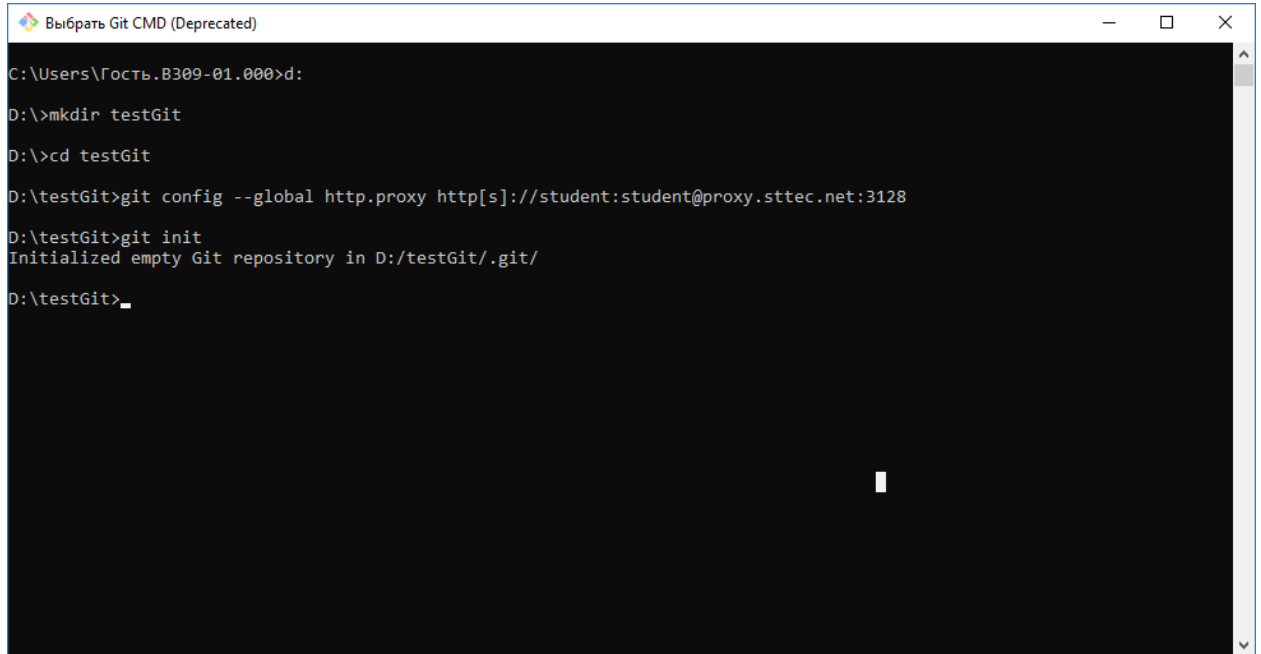
После того как мы оказались в нужной нам папке, мы вводим следующую команду:

`git config --global http.proxy http[s]://student:student@proxy.sttec.net:3128`

Она позволит нам работать с `git` в полном доступе (это необходимо делать только в колледже, потому что тут присутствует прокси, если вы занимаетесь этим дома, то эту операцию производить не нужно).

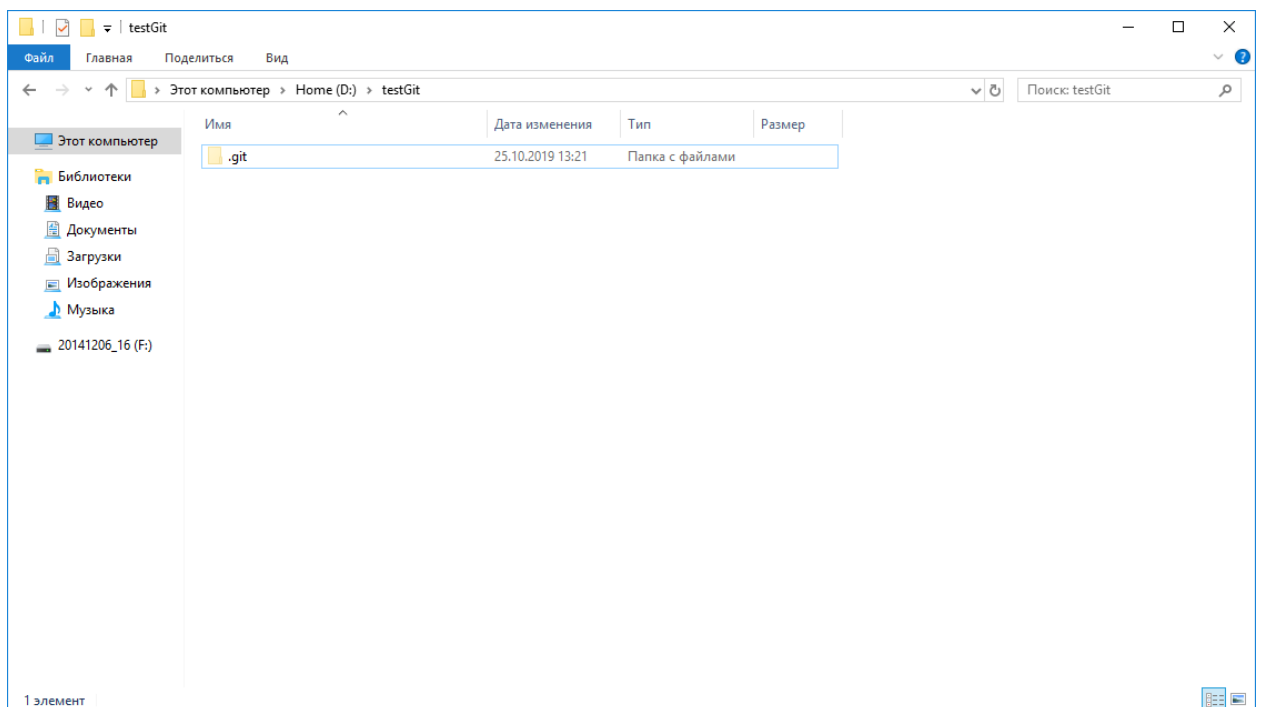
Если вам ничего не написало, значит вы сделали все правильно.

Далее создаем сам репозиторий написав команду **git init**



```
C:\Users\Гость.В309-01.000>d:
D:\>mkdir testGit
D:\>cd testGit
D:\testGit>git config --global http.proxy http[s]://student:student@proxy.sttec.net:3128
D:\testGit>git init
Initialized empty Git repository in D:/testGit/.git/
D:\testGit>
```

Если вы увидели надпись **Initialized empty Git repository in ...**, значит репозиторий успешно создан в вашей папке. В папке появилась скрытая папка под названием **.git**, если же у вас нет этой папки, тогда необходимо нажать **Вид – Параметры – Изменить параметры папок и поиска**, затем перейти во вкладку **Вид** и в самом конце включить: **Показывать скрытые файлы, папки и диски**.



Далее пройдемся по командам:

git status – Показывает в какой ветке вы находитесь (master – это главная ветка репозитория), показывает измененные и добавленные файлы.

Если мы добавим в (!)нашу папку (не в папку .git) текстовый документ, то получим следующие:

```
D:\testGit>git status
On branch master

No commits yet

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

        test.txt

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

Красным подсвечиваются файлы, которые были добавлены, либо изменены.

Далее нам необходимо внести эти изменения в наш репозиторий командой

git add . – добавляет все файлы, содержащиеся в папке в наш репозиторий, точка после add означает, что будут добавлены все файлы, вместо точки можно добавлять конкретные файлы. После того как мы прописали команду **git add .**, далее мы пишем **git status** и там показывается, что были добавлены в репозиторий файлы

```
D:\testGit>git add .

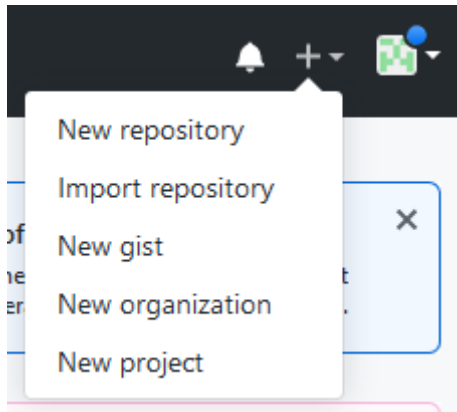
D:\testGit>git status
On branch master

No commits yet

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

        new file:   test.txt
```


Далее, чтоб занести наши файлы в удаленный репозиторий на github нам необходимо зайти на сайт **github.com** в свой аккаунт и в правом верхнем углу нажать на плюс и в всплывающем списке выбрать **new repository**



Нам необходимо ввести название нашего репозитория, его описание (можно не писать) и выбрать каким он будет, публичным или приватным

Create a new repository


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

Owner:  dywud ▾ / Repository name *: testGit ✓

Great repository names are short and memorable. Need inspiration? How about **vigilant-garbanzo**?

Description (optional)

☒  **Public**
Anyone can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer.

Add .gitignore: None ▾

Add a license: None ▾



Create repository

Нажимаем кнопку и наш репозиторий создается, нас перебрасывает на другую страницу и нам нужно будет сохранить вот эти две строки и записать их куда-нибудь:

git remote add origin <https://github.com/dywud/testGit.git>

git push -u origin master

```
git remote add origin https://github.com/ksinaj/test.git
git push -u origin main
```

Наш репозиторий на github создан, про него пока забудем. Вернемся к git CMD. В предыдущих шагах мы добавили в наш локальный репозиторий текстовый документ и проверили это. Следующим нашим шагом будет сделать коммит, коммит – это некий сейв содержимого нашего репозитория, в который мы добавили файлы, с комментарием к этому сейву, по которому мы в дальнейшем сможем находить нужную нам версию условной программы, сайта и т.п. Именно этот коммит мы будем кидать в наш удаленный репозиторий.

Синтаксис коммита:

git commit -m “комментарий”

```
D:\testGit>git commit -m "first commit"
[master (root-commit) a9eaf94] first commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 test.txt
```

Если вы получили данное сообщение, значит коммит создан успешно. После того как мы создали коммит, мы можем закидывать его на удаленный репозиторий. Тут нам потребуются команды, которые мы ранее сохраняли. Прописываем в Git CMD мы пишем следующее (у вас это своя ссылка):

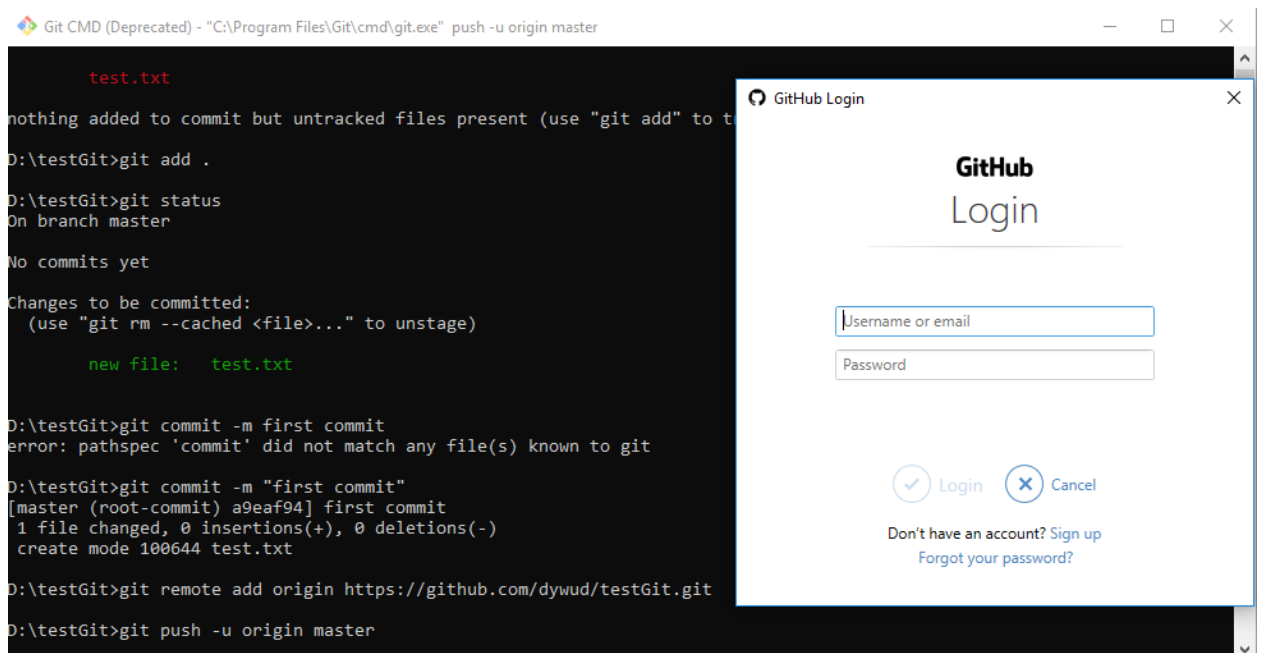
git remote add origin <https://github.com/dywud/testGit.git>

далее:

git push -u origin master

```
git remote add origin https://try.gogs.io/ksinaj/myRepository1.git
git push -u origin master
```

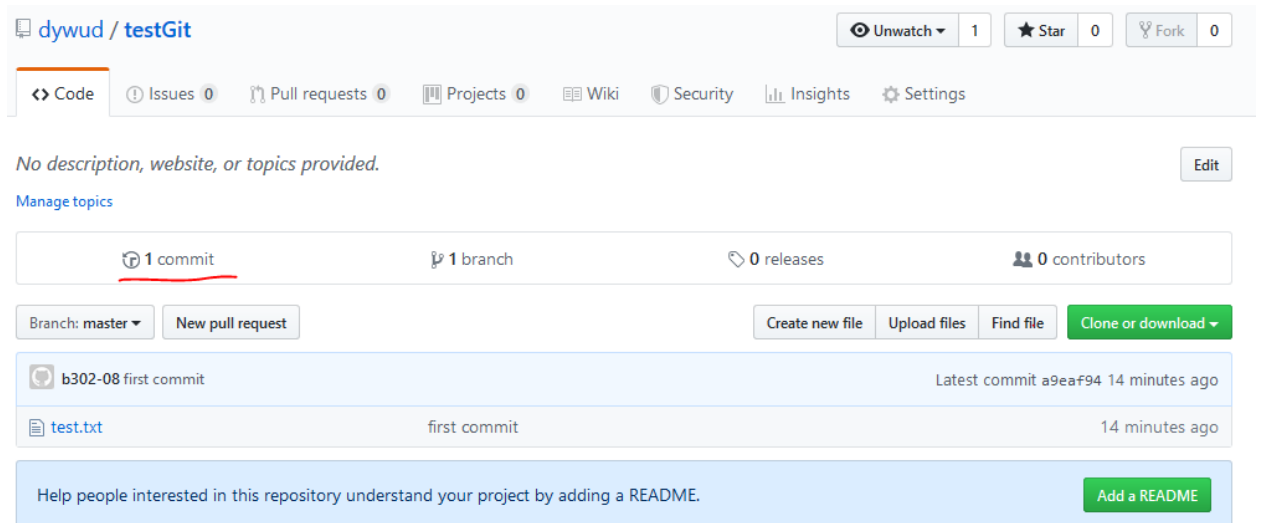
У вас должно всплыть окно с авторизацией в GitHub, если этого не произошло, попробуйте повторить последнюю команду



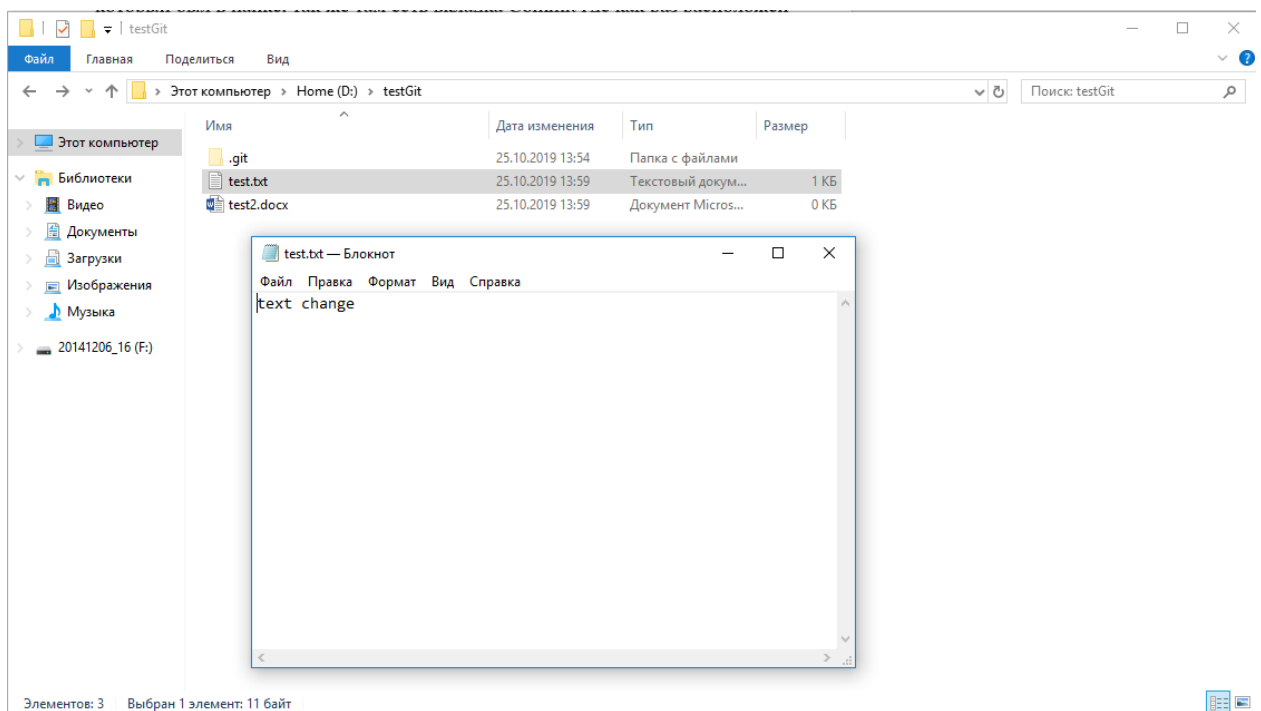
Вводим свой логин и пароль и подключаемся к GitHub, если все получилось правильно, то в Git CMD нам сообщит об ошибке отправки запроса и попросят ввести логин и пароль прямо там, при вводе пароля никакие символы не будут отображаться. Если вы получили следующее сообщение, то ваш сейв перенесен в удаленный репозиторий на GitHub.

```
fatal: HttpRequestException encountered.  
    Произошла ошибка при отправке запроса.  
Username for 'https://github.com': dywud  
Password for 'https://dywud@github.com':  
fatal: UriFormatException encountered.  
    queryUrl  
Enumerating objects: 3, done.  
Counting objects: 100% (3/3), done.  
Writing objects: 100% (3/3), 201 bytes | 201.00 KiB/s, done.  
Total 3 (delta 0), reused 0 (delta 0)  
To https://github.com/dywud/testGit.git  
 * [new branch]      master -> master  
Branch 'master' set up to track remote branch 'master' from 'origin'.
```


Далее переходим на GitHub в наш репозиторий и видим там наш файл, который был в папке, так же там есть вкладка Commit где как раз расположен наш коммит (сейв)



Следующим шагом мы добавим еще один файл в нашу папку и изменим текст в текстовом документе, далее вы поймете для чего.



Совершаем проверку файлов и добавление их в локальный репозиторий

```
D:\testGit>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 checkout -- <file>..." to discard changes in working directory)

        modified:   test.txt

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

        test2.docx

no changes added to commit (use "git add" and/or "git commit -a")
D:\testGit>git add .
```

Тут нам показывает, что текстовый файл был изменен, а также добавлен документ с расширением .docx, мы заносим все эти изменения в наш репозиторий и создаем очередной коммит (сейв) наших изменений.

```
D:\testGit>git commit -m "second commit"
[master e8a1f24] second commit
2 files changed, 1 insertion(+)
create mode 100644 test2.docx
```

Далее заносим этот коммит в наш удаленный репозиторий, но уже из тех двух строк которые мы сохраняли, нам нужно использовать только одну:

git push -u origin master

Первую строку нам необходимо писать только один раз, все сохранения проводятся с помощью строки, которая написана выше. При каждом сохранении нас будут просить ввести логин и пароль от нашего GitHub. Если вы получили следующее, то вы успешно сохранили коммит

```
D:\testGit>git push -u origin master
fatal: HttpRequestException encountered.
    Произошла ошибка при отправке запроса.
Username for 'https://github.com': dywud
Password for 'https://dywud@github.com':
fatal: UriFormatException encountered.
    queryUrl
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 274 bytes | 274.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/dywud/testGit.git
    a9eaf94..e8a1f24  master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Теперь в нашем удаленном репозитории мы видим два файла, число коммитов стало 2, так же синим цветом показано, при каком коммите произошло изменение файла.

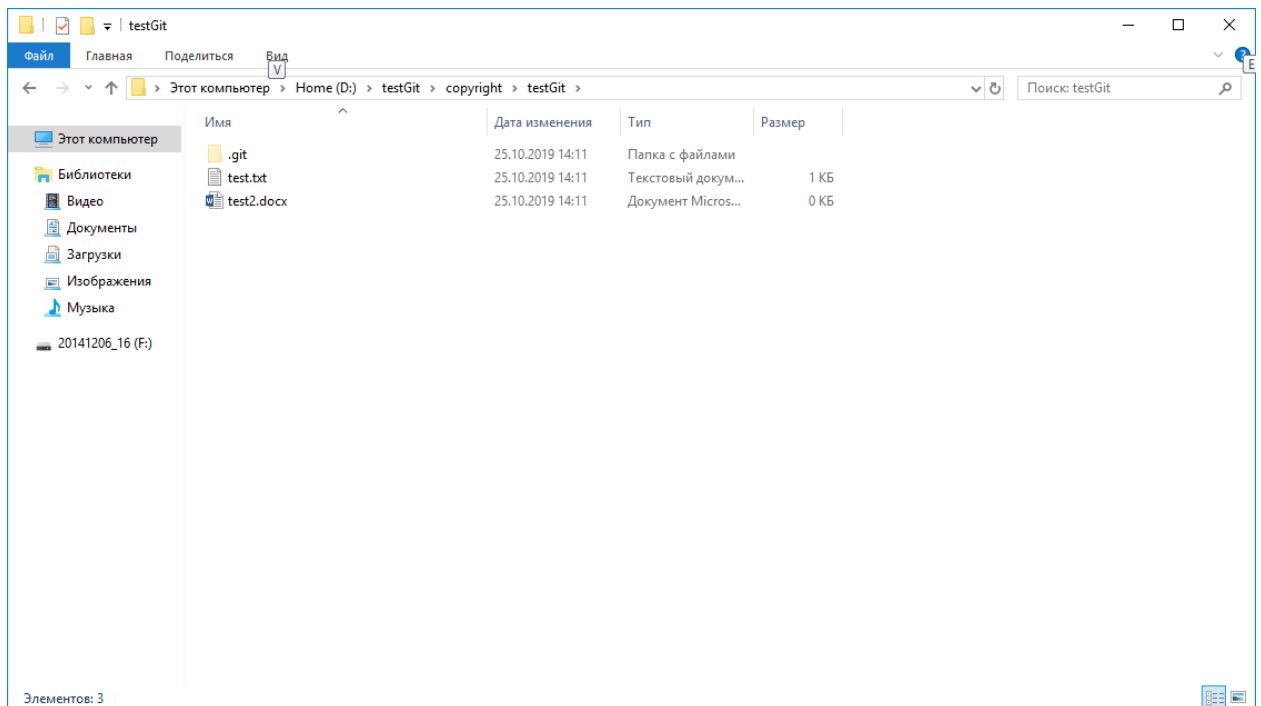
The screenshot shows the GitHub interface for a repository named 'testGit' by user 'dywud'. At the top, there are navigation tabs: 'Code', 'Issues' (0), 'Pull requests' (0), 'Projects' (0), 'Wiki', 'Security', 'Insights', and 'Settings'. Below the repository name, it says 'No description, website, or topics provided.' with an 'Edit' button. A summary bar shows '2 commits' (underlined in red), '1 branch', '0 releases', and '0 contributors'. Below this, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and a green 'Clone or download' button. The commit history shows two commits, both labeled 'second commit' (underlined in blue). The first commit is 'b302-08' and the second is 'e8a1f24', both from 4 minutes ago. The files listed are 'test.txt' and 'test2.docx'. At the bottom, there is a blue box with the text 'Help people interested in this repository understand your project by adding a README.' and a green 'Add a README' button.

Для начала покажу как из уже имеющегося удаленного репозитория копировать файлы к себе на ПК. Переходим в нужную папку, где мы хотим видеть наши файлы и прописываем след. команду:

git clone ссылка_на_ваш_удаленный_репозиторий

```
D:\testGit\copyright>git clone https://github.com/dywud/testGit
Cloning into 'testGit'...
fatal: UriFormatException encountered.
  queryUrl
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 0), reused 6 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), done.
```

И теперь в нашей папке появилась папка с названием нашего репозитория и в этой папке содержатся все те же файлы, которые у нас были там.



Теперь перейдем к самому интересному, вернемся к нашему репозиторию

Прописываем команду **git log**, там показаны все сейвы которые были сделаны, и их так называемые коды. Коммиты расположены в порядке убывания, снизу находятся ваши первые сейвы. Если вам нужно отменить коммит, который вы совершили, вернуться к прежней версии вашего проекта, то копируем код нужного вам сейва и пишем следующее:

git checkout код_коммита

```
Git CMD (Deprecated)
D:\testGit\copyright>cd ..

D:\testGit>git log
commit e8a1f24d55d977cef2ce808e28f09a6b6e94a8cc (HEAD -> master, origin/master)
Author: b302-08 <b302-08@gmail.com>
Date:   Fri Oct 25 14:01:56 2019 +0300

    second commit

commit a9eaf94e9c1e3d3f3846e3c960ac6777055a0231
Author: b302-08 <b302-08@gmail.com>
Date:   Fri Oct 25 13:42:39 2019 +0300

    first commit

D:\testGit>git checkout a9eaf94e9c1e3d3f3846e3c960ac6777055a0231
Note: checking out 'a9eaf94e9c1e3d3f3846e3c960ac6777055a0231'.

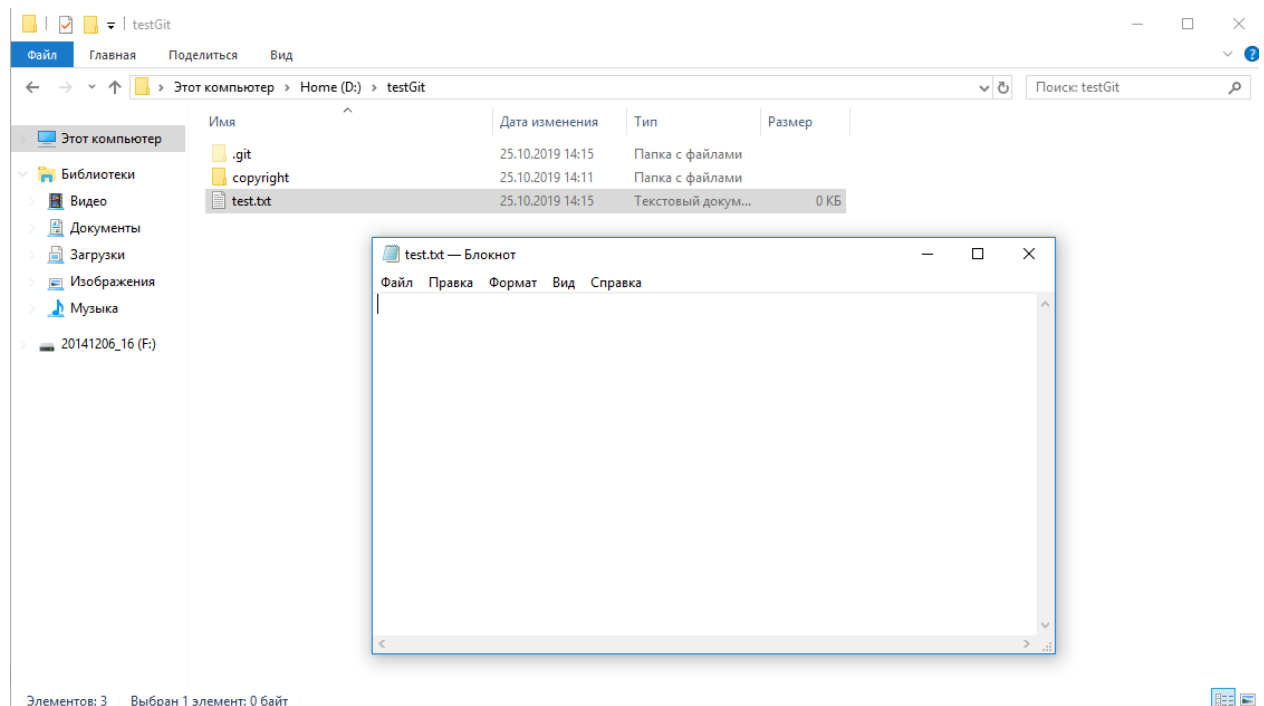
You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

    git checkout -b <new-branch-name>

HEAD is now at a9eaf94 first commit
D:\testGit>
```

Как вы помните, первый коммит у меня был с пустым текстовым файлом, если зайти в папку, то увидим содержимое папки при первом сохранении



Таким образом вы можете переходить между своими коммитами и отменять какие-либо изменения.

Если при очередном сохранении вылезла ошибка вида:

```
D:\testGit>git push -u origin master
fatal: HttpRequestException encountered.
  произошла ошибка при отправке запроса.
Username for 'https://github.com': dywud
Password for 'https://dywud@github.com':
fatal: UriFormatException encountered.
  queryUrl
Everything up-to-date
Branch 'master' set up to track remote branch 'master' from 'origin'.
```







Это означает, что вы находитесь не на ветке мастер, ветки можно проверить командой **git branch**, * означает то, на какой ветке вы сейчас находитесь, если эта звездочка стоит не возле ветки **master**, тогда нужно прописать команду **git checkout master**

```
D:\testGit>git branch
* (HEAD detached at e8a1f24)
master
```

```
D:\testGit>git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

D:\testGit>git branch
* master
```

Если вы в GitHub перейдете в свои коммиты и нажмете на код коммита

third commit b302-08 committed 1 minute ago	 751ac50 
second commit b302-08 committed 24 minutes ago	 <u>e8a1f24</u> 
first commit b302-08 committed 43 minutes ago	 a9eaf94 

То там будет показано какие изменения были произведены, какие файлы добавлены, что было удалено.

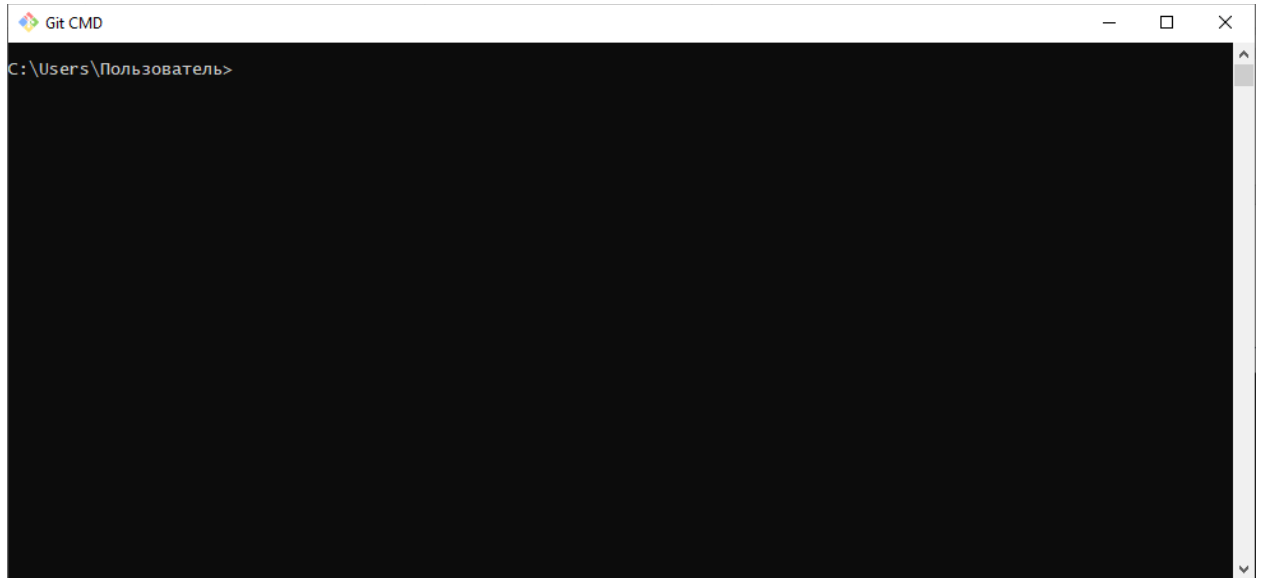
second commit		Browse files
b302-08 committed 25 minutes ago		
1 parent a9eaf94 commit e8a1f24d55d977cef2ce808e28f09a6b6e94a8cc		
Showing 2 changed files with 1 addition and 0 deletions.		
<div>UnifiedSplit</div>		
<div><div>1 test.txt</div><div>... @@ -0,0 +1 @@</div><div>1 + text change</div></div>		
<div><div>0 test2.docx</div><div>No changes.</div></div>		

Руководство по использованию gogs

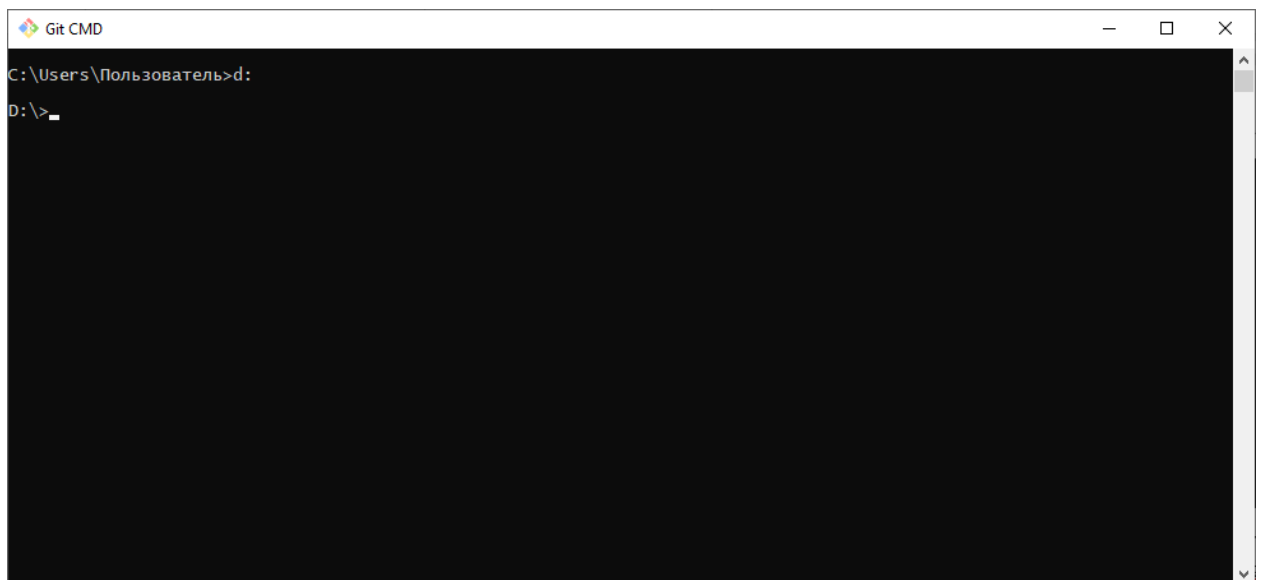
Для начала необходимо зайти на сайт try.gogs.io и зарегистрироваться.

Для работы с git необходимо открыть меню Пуск и ввести Git CMD, открываем его.

Так выглядит его интерфейс:



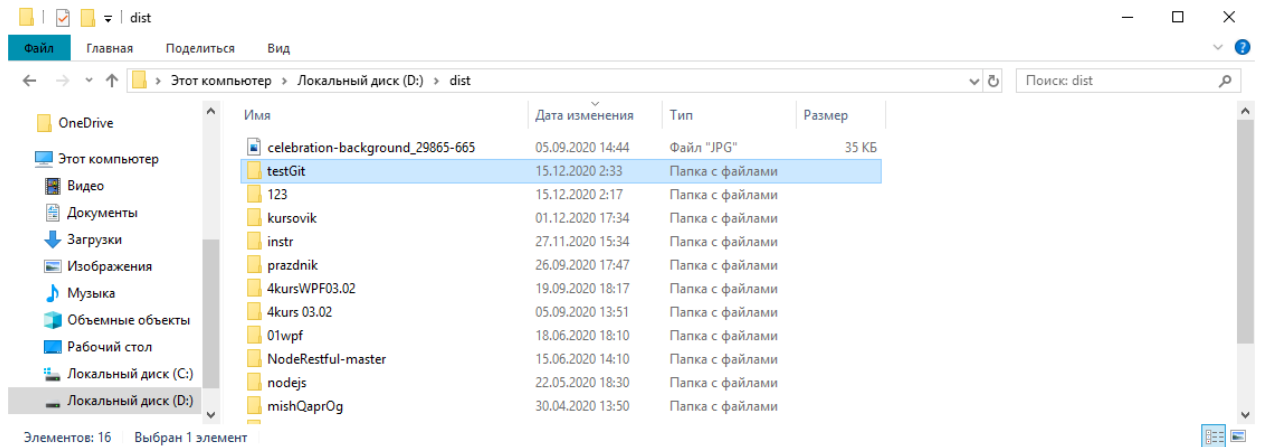
Далее нам необходимо перейти в нужную нам папку где мы будем создавать репозиторий. Переход осуществляется путем ввода в git команду **cd путь:**, для перехода на уровень ниже, необходимо написать **cd ..** для перехода на другой диск можно просто написать **d:**



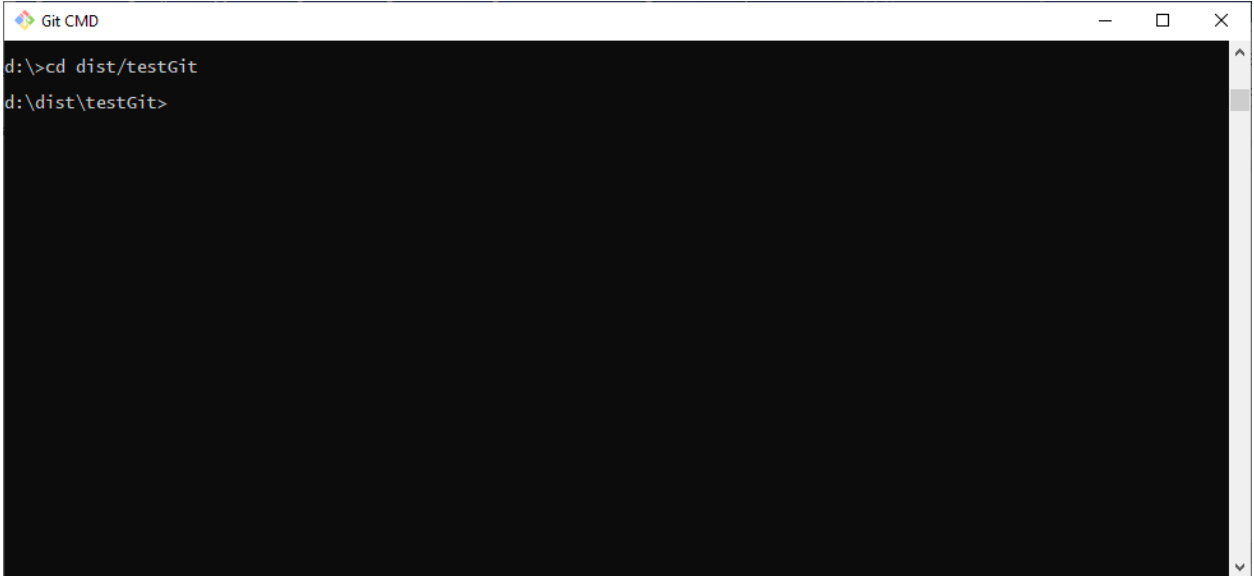
Если же вам необходимо создать папку в которой будет находиться репозиторий, тогда необходимо прописать: **mkdir название_папки**, в моем случае я создам папку с названием testGit

```
d:\dist>mkdir testGit
```

После этого на диске D: появится папка.



Командой `cd название_папки` мы переходим непосредственно в нашу папку

A screenshot of a terminal window titled "Git CMD". The window has a black background with white text. The first line shows the command `d:\>cd dist/testGit`. The second line shows the prompt `d:\dist\testGit>`, indicating the command was executed successfully and the current directory is now `dist\testGit`.

```
Git CMD
d:\>cd dist/testGit
d:\dist\testGit>
```

Если вам необходимо создать репозиторий уже в имеющейся папке, то вам необходимо просто в нее перейти, создавать ничего не нужно.

Репозиторий – это место, где хранится и поддерживается какие-либо данные.

После того как мы оказались в нужной нам папке, мы вводим следующую команду:

`git config --global http.proxy http[s]://student:student@proxy.sttec.net:3128`

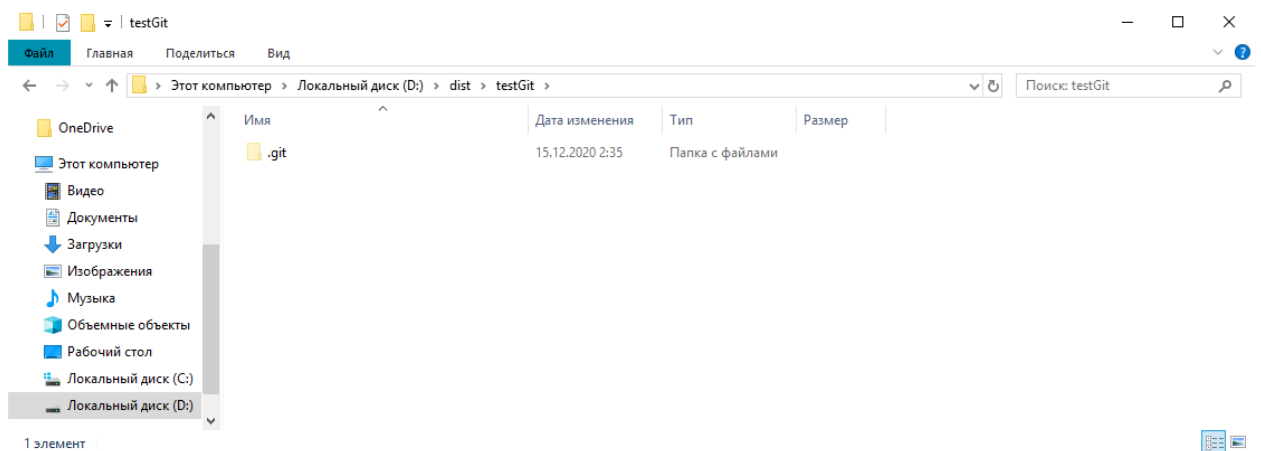
Она позволит нам работать с git в полном доступе (это необходимо делать только в колледже, потому что тут присутствует прокси, если вы занимаетесь этим дома, то эту операцию производить не нужно).

Если вам ничего не написало, значит вы сделали все правильно.

Далее создаем сам репозиторий написав команду **git init**

```
Git CMD
d:\dist>mkdir testGit
d:\dist>cd testGit
d:\dist\testGit>git init
Initialized empty Git repository in D:/dist/testGit/.git/
d:\dist\testGit>
```

Если вы увидели надпись **Initialized empty Git repository in ...**, значит репозиторий успешно создан в вашей папке. В папке появилась скрытая папка под названием **.git**, если же у вас нет этой папки, тогда необходимо нажать **Вид – Параметры – Изменить параметры папок и поиска**, затем перейти во вкладку **Вид** и в самом конце включить: **Показывать скрытые файлы, папки и диски**.



Далее пройдемся по командам:

git status – Показывает в какой ветке вы находитесь (master – это главная ветка репозитория), показывает измененные и добавленные файлы.

Если мы добавим в (!)нашу папку (не в папку .git) текстовый документ, то получим следующие:

```
d:\dist\testGit>git status
On branch master

No commits yet

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

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

Красным подсвечиваются файлы, которые были добавлены, либо изменены.

Далее нам необходимо внести эти изменения в наш репозиторий командой

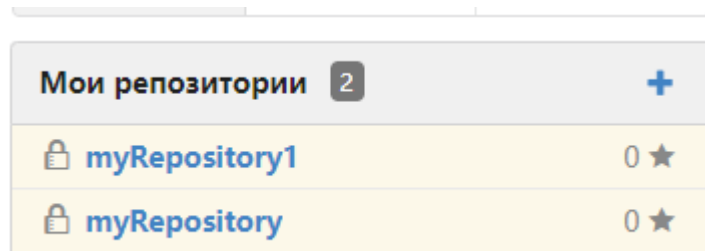
git add . – добавляет все файлы, содержащиеся в папке в наш репозиторий, точка после add означает, что будут добавлены все файлы, вместо точки можно добавлять конкретные файлы. После того как мы прописали команду **git add .**, далее мы пишем **git status** и там показывается, что были добавлены в репозиторий файлы

```
d:\dist\testGit>git add .
d:\dist\testGit>git status
On branch master

No commits yet

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


Далее, чтоб занести наши файлы в удаленный репозиторий на gogs нам необходимо зайти на сайт **try.gogs.io** в свой аккаунт и во вкладке «Мои репозитории» нажать на плюсики



Нам необходимо ввести название нашего репозитория, его описание (можно не писать) и выбрать каким он будет, публичным или приватным

Новый репозиторий

Владелец *

 ksinaj

Имя репозитория *

myRepository

Лучшие названия репозитория коротки, запоминаемы и уникальны.

Видимость

☒ Личный репозиторий

☐ This repository is Unlisted

Описание

Описание репозитория. Максимальная длина 512 символов.

Доступные символы: 512

.gitignore

Выберите шаблоны .gitignore

Лицензия

Выберите файл лицензии

Readme ?

Default

☐ Инициализировать этот репозиторий выбранными файлами и шаблоном

Создать репозиторий

Отмена

Нажимаем кнопку и наш репозиторий создается, нас перебрасывает на другую страницу и нам нужно будет сохранить вот эти две строки и записать их куда-нибудь:

```
git remote add origin https://try.gogs.io/ksinaj/testGit.git  
git push -u origin master
```

Наш репозиторий на gogs создан, про него пока забудем. Вернемся к git CMD. В предыдущих шагах мы добавили в наш локальный репозиторий текстовый документ и проверили это. Следующим нашим шагом будет сделать коммит, коммит – это некий сейв содержимого нашего репозитория, в который мы добавили файлы, с комментарием к этому сейву, по которому мы в дальнейшем сможем находить нужную нам версию условной программы, сайта и т.п. Именно этот коммит мы будем кидать в наш удаленный репозиторий.

Синтаксис коммита:

git commit -m “комментарий”

```
d:\dist\testGit>git commit -m "first commit"
[master (root-commit) fle8c6e] first commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 test.txt
```

Если вы получили данное сообщение, значит коммит создан успешно. После того как мы создали коммит, мы можем закидывать его на удаленный репозиторий. Тут нам потребуются команды, которые мы ранее сохраняли. Прописываем в Git CMD мы пишем следующее (у вас это своя ссылка):

git remote add origin <https://github.com/ksinaj/test.git>

далее:


git push -u origin master




У вас должно всплыть окно с авторизацией в gogs, если этого не произошло, попробуйте повторить последнюю команду (окно с авторизацией вылезает лишь при первом занесении файлов)






Вводим свой логин и пароль и подключаемся к gogs

```
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (9/9), 619 bytes | 619.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
To https://try.gogs.io/ksinaj/testGit.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```




Далее переходим на gogs в наш репозиторий и видим там наш файл, который был в папке, так же там есть вкладка Коммиты, где как раз расположен наш коммит (сейв)








 **ksinaj / testGit**


 Перестать следить **1**  В избранное **0**  Ответить **0**


 **Файлы**  Задачи **0**  Запросы на слияние **0**  Вики  Настройки

Нет описания

 **1** Коммиты  **1** Ветки  **0** Релизы

 Ветка: **master** **testGit**  Новый файл  Загрузить файл  HTTPS  SSH <https://try.gogs.io/ksinaj/testGit>  

 **My Name** **f1e8c6e821** first commit 7 минут назад

 **test.txt** **f1e8c6e821** first commit 7 минут назад

Readme файлы

Файл `readme.md` — это основа любого проекта с открытым исходным кодом. Он дает полное понимание того, куда движется проект. Он объясняет, что это за ПО и зачем оно нужно. В нем указаны предварительные условия, которые помогают новым участникам проекта быстрее включиться в работу.

Самое важное: в `readme`-файле сказано, как запустить это ПО с целью разработки. Также в `readme` обязательно должна быть инструкция по развертыванию ПО в среде эксплуатации.

Что нужно включить в файл README?

Так что же должен содержать идеальный файл README? В качестве отправной точки рекомендуется воспользоваться следующим списком:

1. Название продукта

Не забудьте дать своему проекту имя. На GitHub просто на удивление много безымянных проектов.

2. Введение или краткое описание

Напишите две-три короткие строчки, поясняющие, что делает ваш проект и для кого он предназначен. Не вставляйте заголовки типа «Вступление», «Обзор» и т. п. — и так очевидно, что это введение.

3. Необходимые условия для использования продукта

Сразу после введения добавьте раздел, где будут перечислены все знания и инструменты, необходимые тому, кто пожелает воспользоваться вашим проектом. Например, если продукт запускается на последней версии Python, напишите, что нужно установить Python.

4. Как установить программу

Опишите шаги инсталляции! Просто поразительно, сколько есть проектов, где описано, как использовать продукт, но нет ни слова о том, как его установить. Вероятно, ожидается, что читатель волшебным образом сам догадается. Если процесс установки достаточно длинный (сложный), обязательно разбейте его на отдельные этапы и пронумеруйте их.

5. Порядок использования

Опишите, как пользователь может использовать ваш проект после установки. Обязательно включите примеры использования, ссылки на пояснение опций команд или флагов (если считаете, что это будет полезно). Если у вас есть более подробная документация в отдельном файле или на сайте, поставьте ссылку на нее.

6. Как принять участие в проекте

Опишите шаги, которые должен пройти потенциальный контрибьютор. Возможно, вы могли бы создать руководство в отдельном файле и поместить ссылку на него в README. Укажите в руководстве все, что вы хотите, чтобы люди знали, прежде чем отправлять пул-реквесты.

7. Добавьте список контрибьюторов

Укажите всех людей, которые участвовали в создании этого проекта. Это хороший способ сделать так, чтобы open source казался плодом командных усилий. Также таким образом вы поблагодарите всех людей, потративших время на участие в вашем проекте.

8. Добавьте раздел с благодарностями

Также, если вы использовали чью-то еще работу (код, дизайн, изображения и т. п.) и копирайт обязывает вас указать автора, вы можете сделать это, добавив специальный раздел. Тут можно поблагодарить и других разработчиков или даже целые организации, оказавшие помощь проекту.

9. Контактная информация

Возможно, вы замкнутый человек, избегающий любой публичности, и совершенно не хотите раскрывать свои контакты. Но лучше все же их добавить где-нибудь на видном месте — на случай, если у людей возникнут вопросы по продукту, если кто-то захочет принять участие в разработке или — чем черт не шутит! — если кто-то так восхитится вашим проектом, что захочет предложить вам работу.

10. Информация о лицензии

Информацию о лицензии продукта определенно стоит включить в файл README. Стартапы и прочие компании, использующие стороннее ПО, не смогут использовать ваш продукт, если не будут знать, на каких условиях могут это делать.

Добавьте немного блеска

Если хотите, чтобы ваш README выделялся и имел привлекательный вид —

- **Добавьте логотип.** Если у вашего проекта есть лого, разместите его в верхней части README. Благодаря брендингу проект выглядит более профессиональным, кроме того, это помогает людям его запомнить.
- **Добавьте значки или плашки.** Они помогут вам быстро показать посетителям текущий статус проекта, лицензию, обновление зависимостей. Плюс, они просто круто выглядят!
- **Добавьте скриншоты.** Иногда простой скриншот или серия скриншотов бывают полезнее тысячи слов. Но будьте внимательны! Если вы используете скриншоты, следите за их актуальностью и обновляйте по мере обновления проекта.
- **Используйте эмодзи(?).** Сегодня во многих проектах используются эмодзи, но, конечно, только от вас зависит, хотите ли вы тоже их использовать. Это может быть хорошим способом добавить немного цвета и юмора в ваш README и слегка «очеловечить» проект.