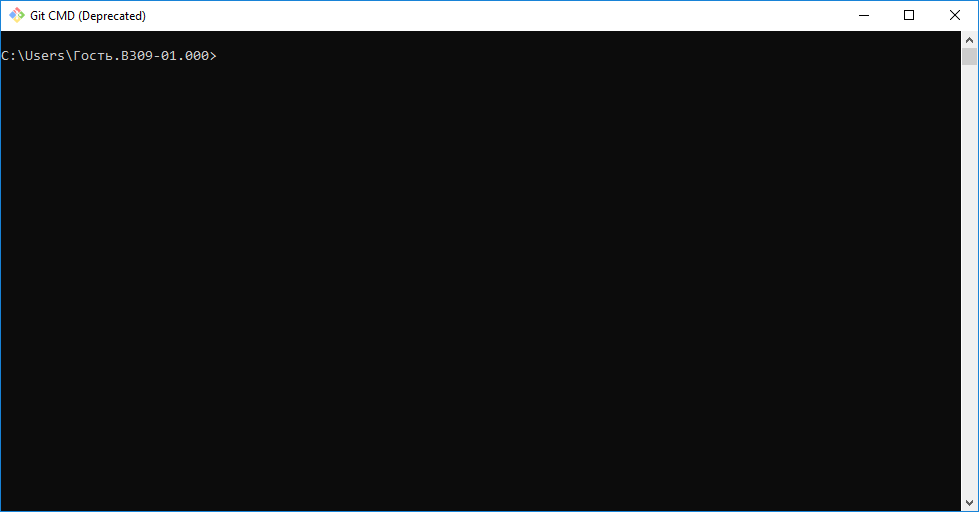
**Руководство по использованию github + git**

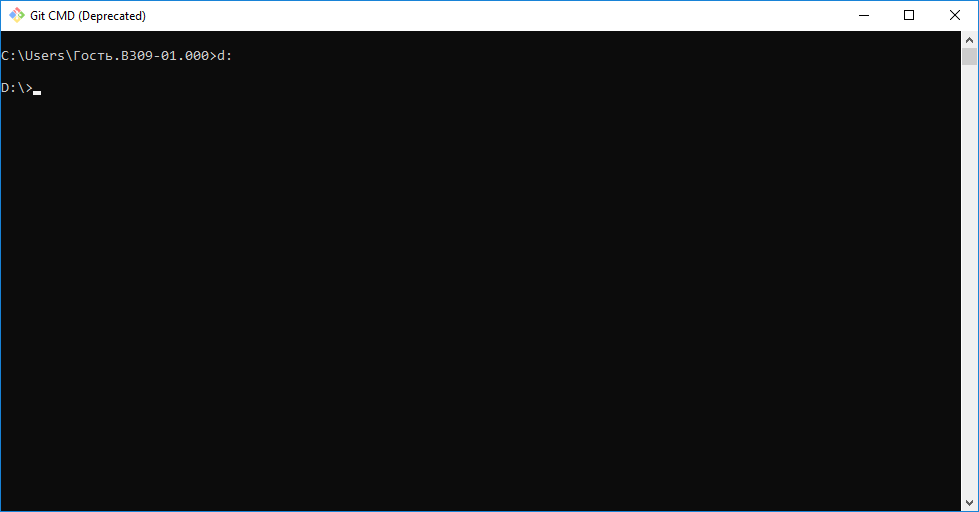
Для начала необходимо зайти на сайт github.com и зарегистрироваться.

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

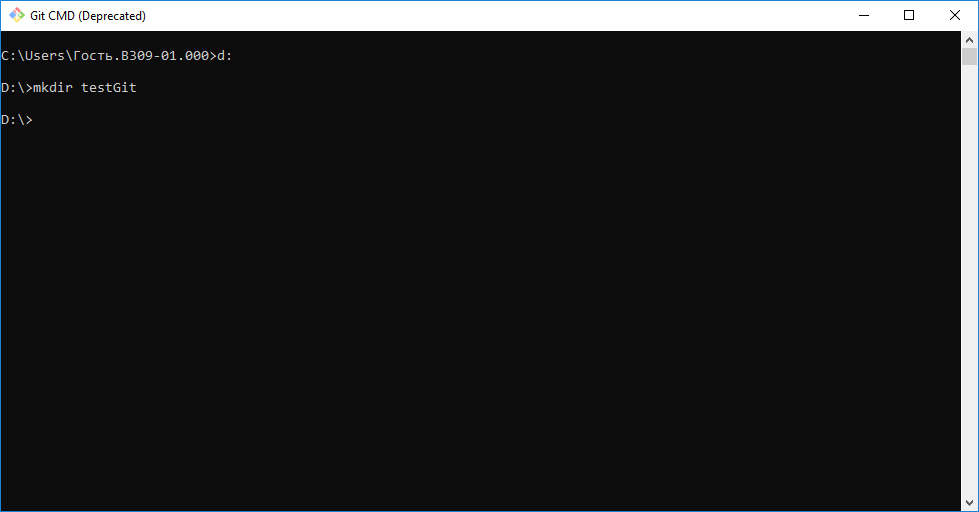
Так выглядит его интерфейс:



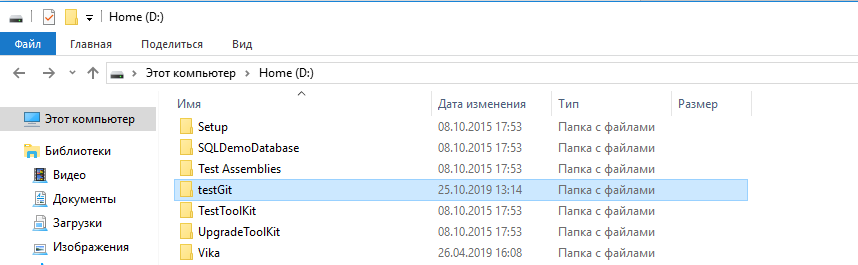
Далее нам необходимо перейти в нужную нам папку где мы будем создавать репозиторий. Переход осуществляется путем ввода в git команду **cd путь:**, для перехода на уровень ниже, необходимо написать **cd ..** для перехода на другой диск можно просто написать **d:**



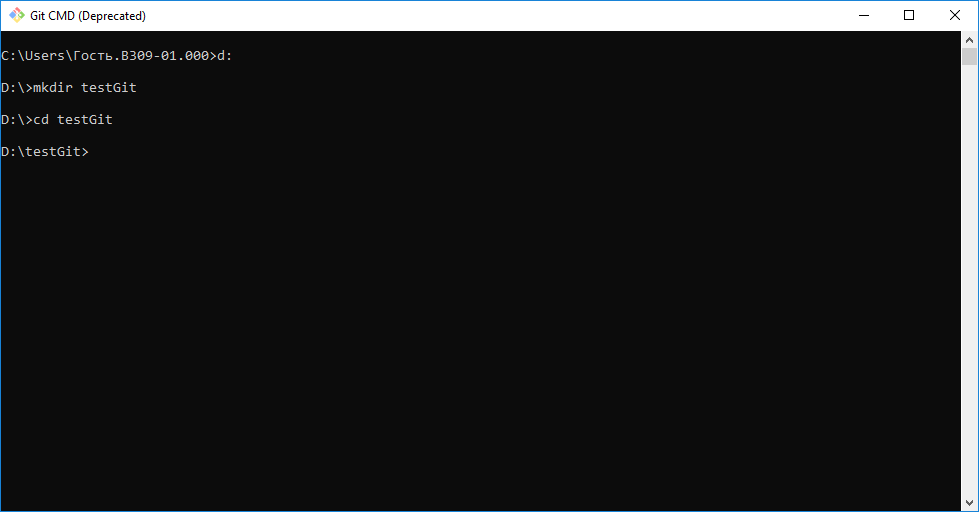
Если же вам необходимо создать папку в которой будет находиться репозиторий, тогда необходимо прописать: **mkdir название\_папки**, в моем случае я создам папку с названием testGit



После этого на диске D: появится папка.



Командой cd название\_папки мы переходим непосредственно в нашу папку



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.

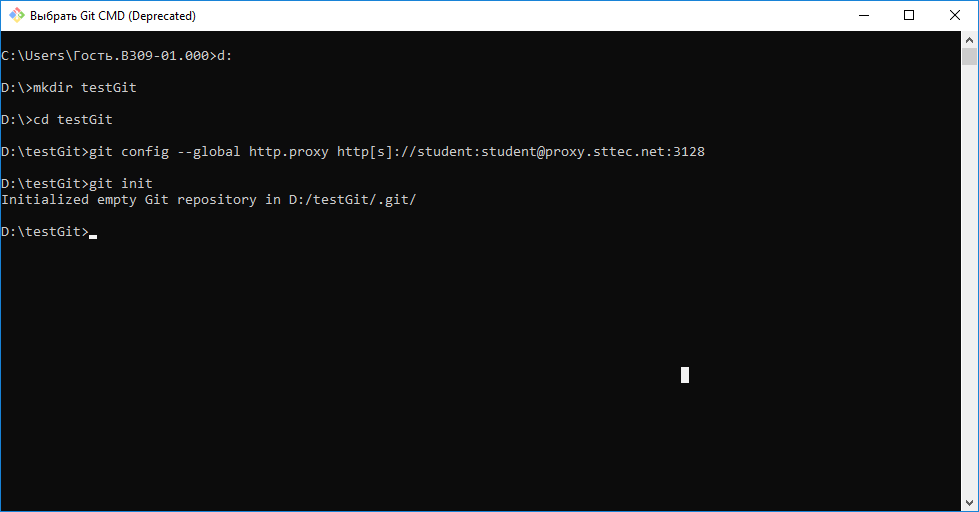
После того как мы оказались в нужной нам папке, мы вводим следующую команду:

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

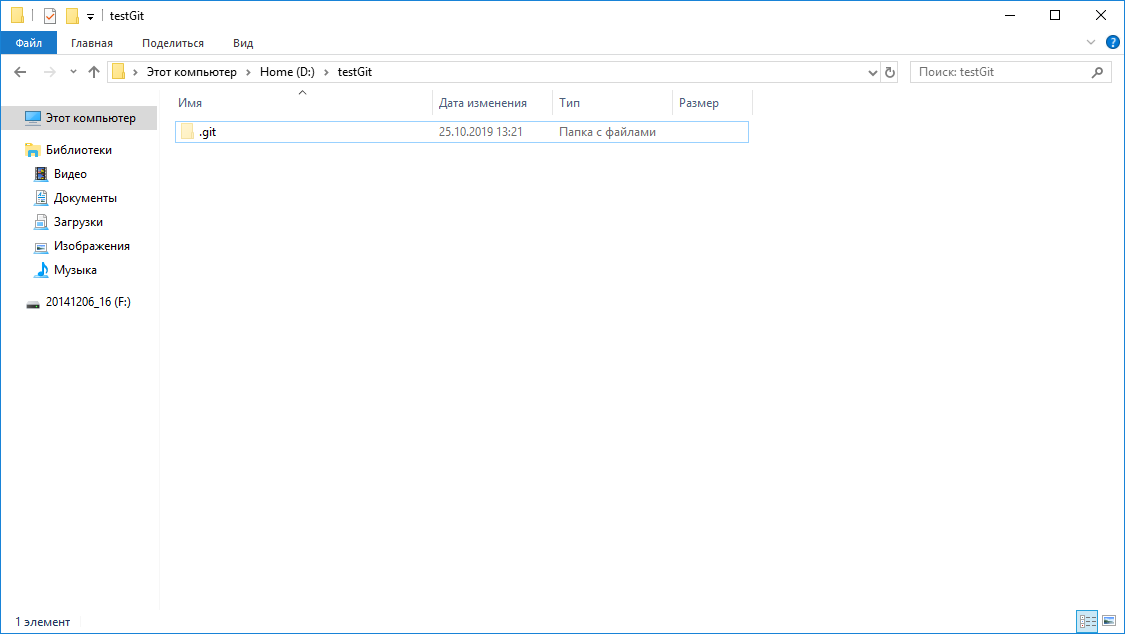
Она позволит нам работать с git в полном доступе (это необходимо делать только в колледже, потому что тут присутствует прокси, если вы занимаетесь этим дома, то эту операцию производить не нужно).

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

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



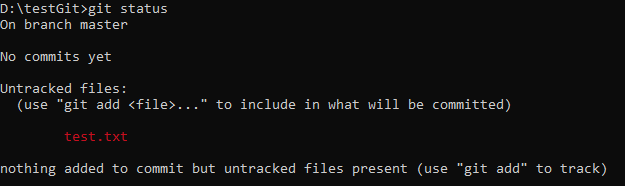
Если вы увидели надпись **Initialized empty Git repository in …**, значит репозиторий успешно создан в вашей папке. В папке появилась скрытая папка под названием **.git**, если же у вас нет этой папки, тогда необходимо нажать **Вид – Параметры – Изменить параметры папок и поиска**, затем перейти во вкладку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

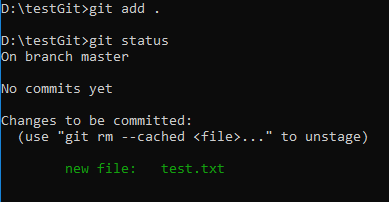
Если мы добавим в (!)нашу папку (не в папку .git) текстовый документ, то получим следующие:



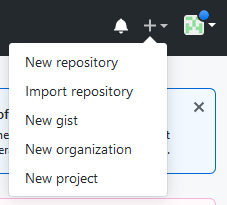
Красным подсвечиваются файлы, которые были добавлены, либо изменены.

Далее нам необходимо внести эти изменения в наш репозиторий командой

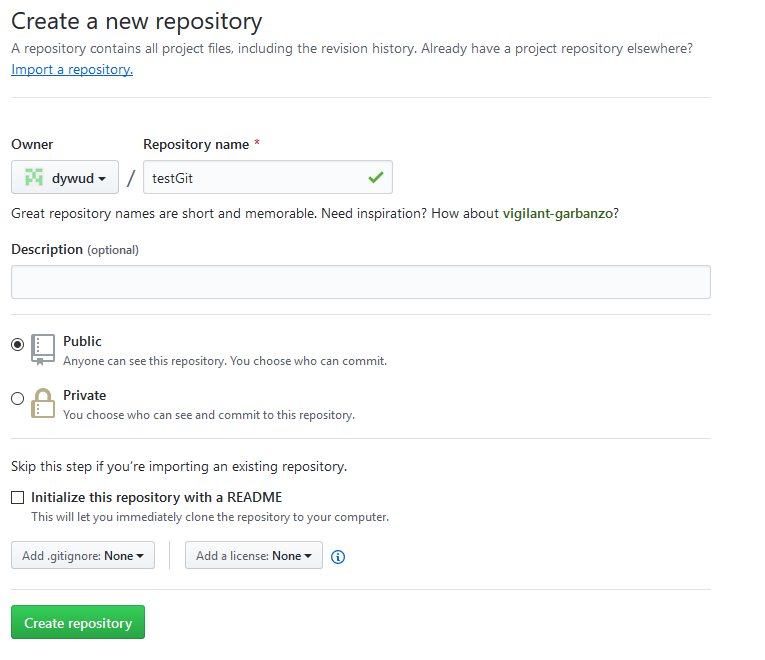
**git add . –** добавляет все файлы, содержащиеся в папке в наш репозиторий, точка после add означает, что будут добавлены все файлы, вместо точки можно добавлять конкретные файлы. После того как мы прописали команду **git add .**, далее мы пишем **git status** и там показывается, что были добавлены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



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



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**

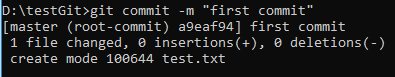
git remote add origin https://github.com/ksinaj/test.git

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”**



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**](https://github.com/dywud/testGit.git)

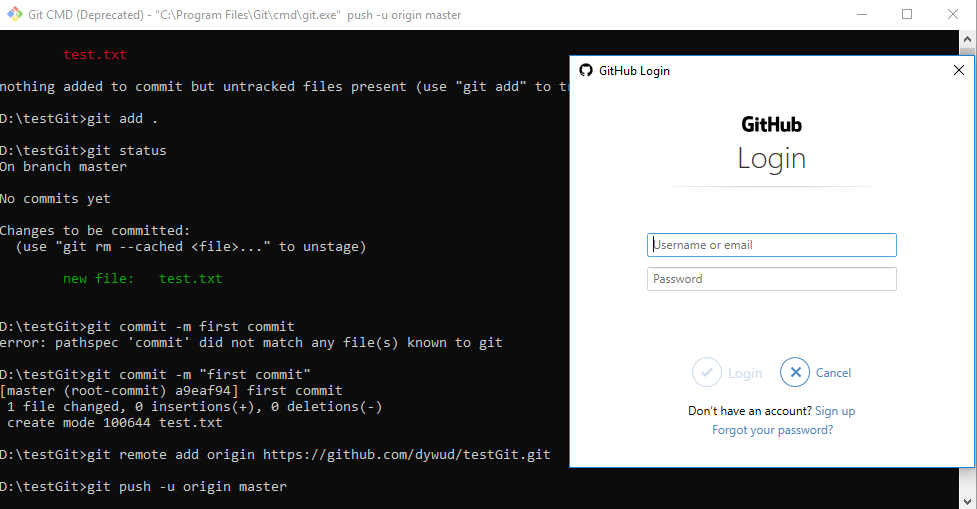
далее:

**git push -u origin master**

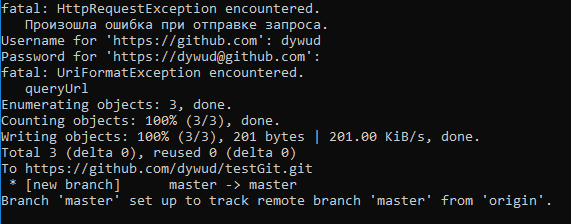
git remote add origin https://try.gogs.io/ksinaj/myRepository1.git

git push -u origin master

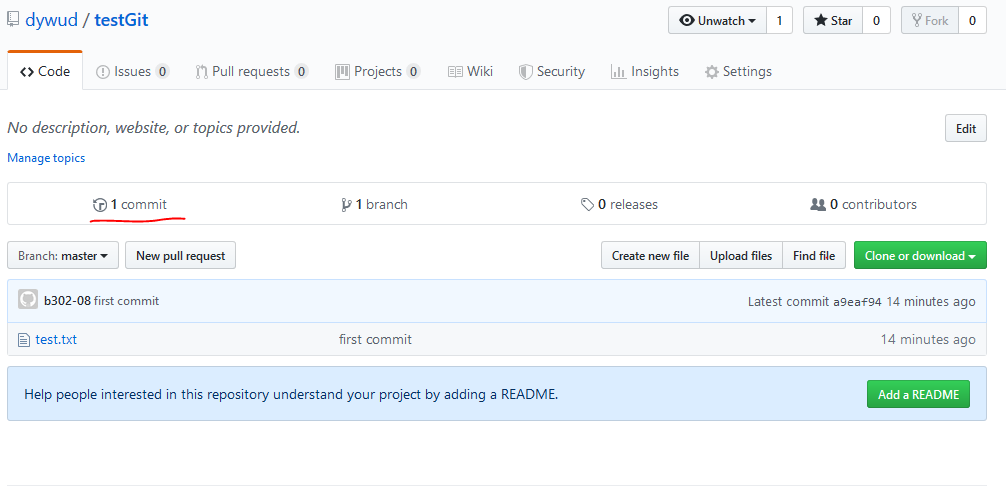
У вас должно всплыть окно с авторизацией в GitHub, если этого не произошло, попробуйте повторить последнюю команду



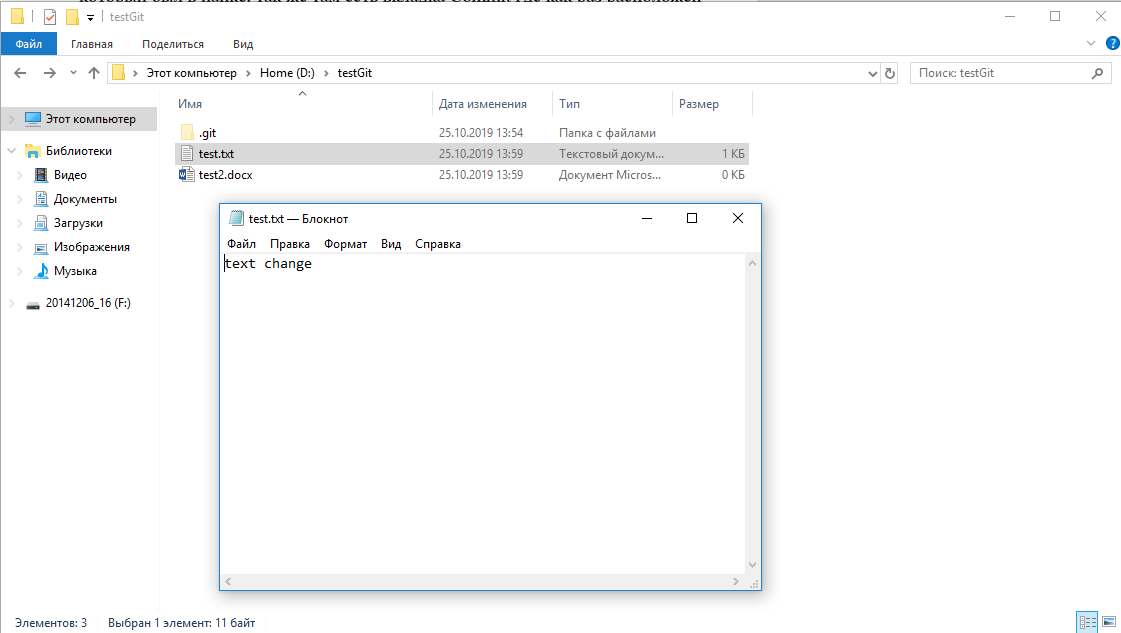
Вводим свой логин и пароль и подключаемся к GitHub, если все получилось правильно, то в Git CMD нам сообщит об ошибке отправки запроса и попросят ввести логин и пароль прямо там, при вводе пароля никакие символы не будут отображаться. Если вы получили следующее сообщение, то ваш сейв перенесен в удаленный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



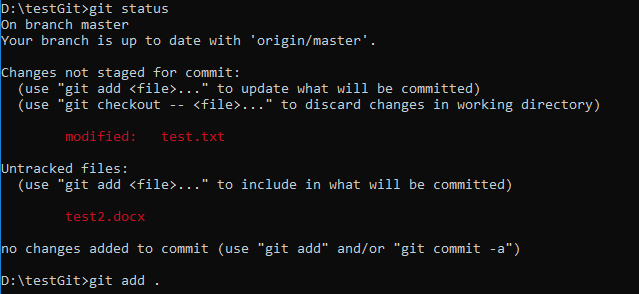
Далее переходим на GitHub в наш репозиторий и видим там наш файл, который был в папке, так же там есть вкладка Commit где как раз расположен наш коммит (сейв)



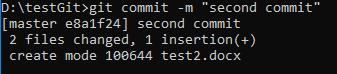
Следующим шагом мы добавим еще один файл в нашу папку и изменим текст в текстовом документе, далее вы поймете для чего.



Совершаем проверку файлов и добавление их в локальный репозиторий

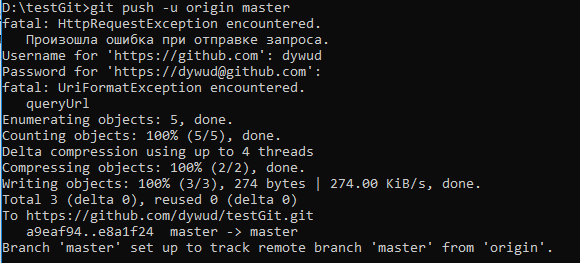


Тут нам показывает, что текстовый файл был изменен, а также добавлен документ с расширением .docx, мы заносим все эти изменения в наш репозиторий и создаем очередной коммит (сейв) наших изменений.

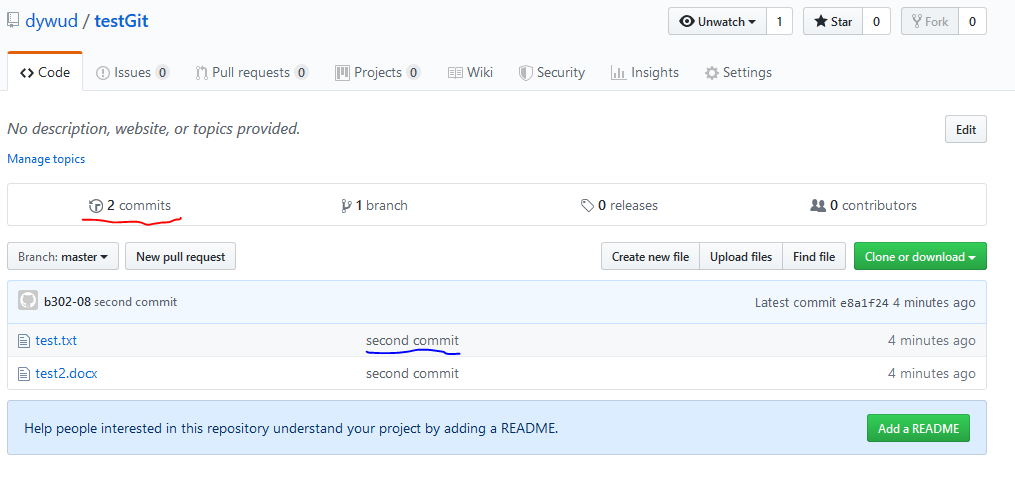


Далее заносим этот коммит в наш удаленный репозиторий, но уже из тех двух строк которые мы сохраняли, нам нужно использовать только одну:

**git push -u origin master**

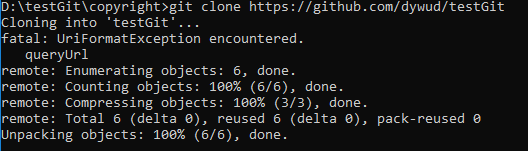
Первую строку нам необходимо писать только один раз, все сохранения проводятся с помощью строки, которая написана выше. При каждом сохранении нас будут просить ввести логин и пароль от нашего GitHub. Если вы получили следующее, то вы успешно сохранили коммит 

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

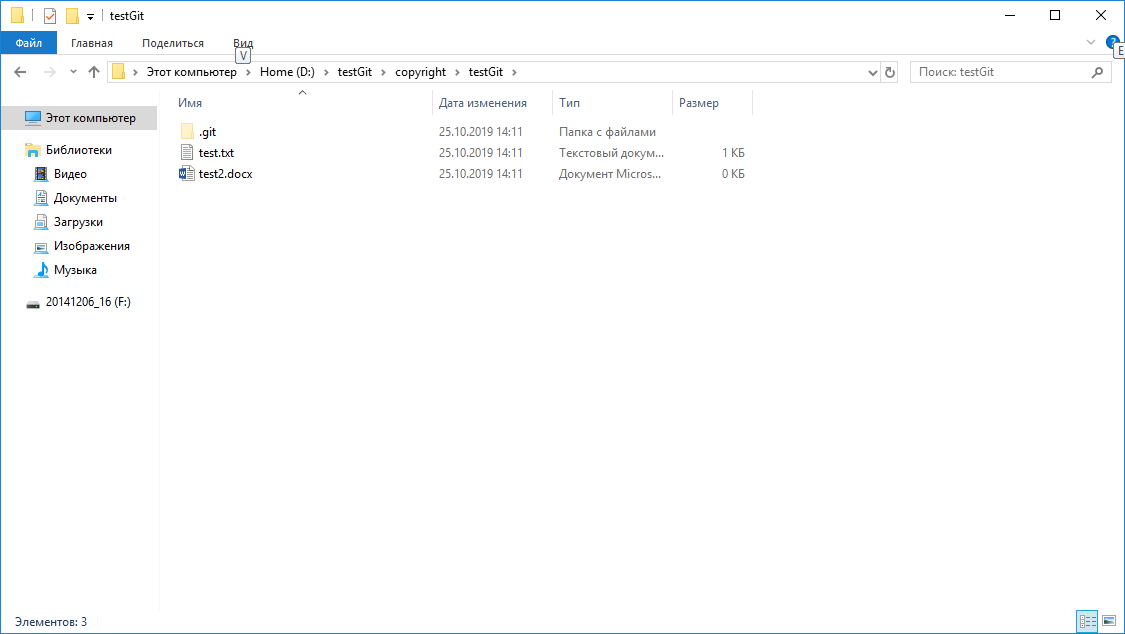


Для начала покажу как из уже имеющегося удаленного репозитория копировать файлы к себе на ПК. Переходим в нужную папку, где мы хотим видеть наши файлы и прописываем след. команду:

**git clone ссылка\_на\_ваш\_удаленный\_репозиторий**



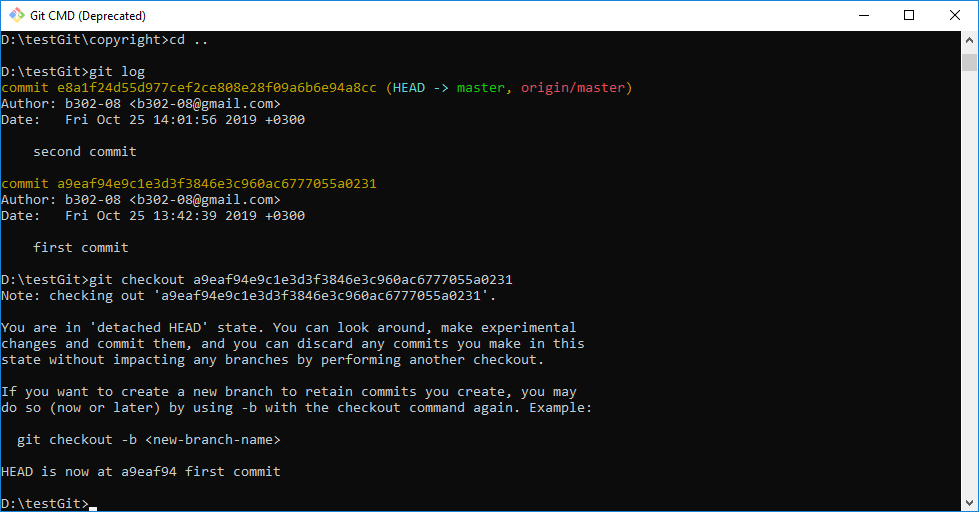
И теперь в нашей папке появилась папка с названием нашего репозитория и в этой папке содержатся все те же файлы, которые у нас были там.



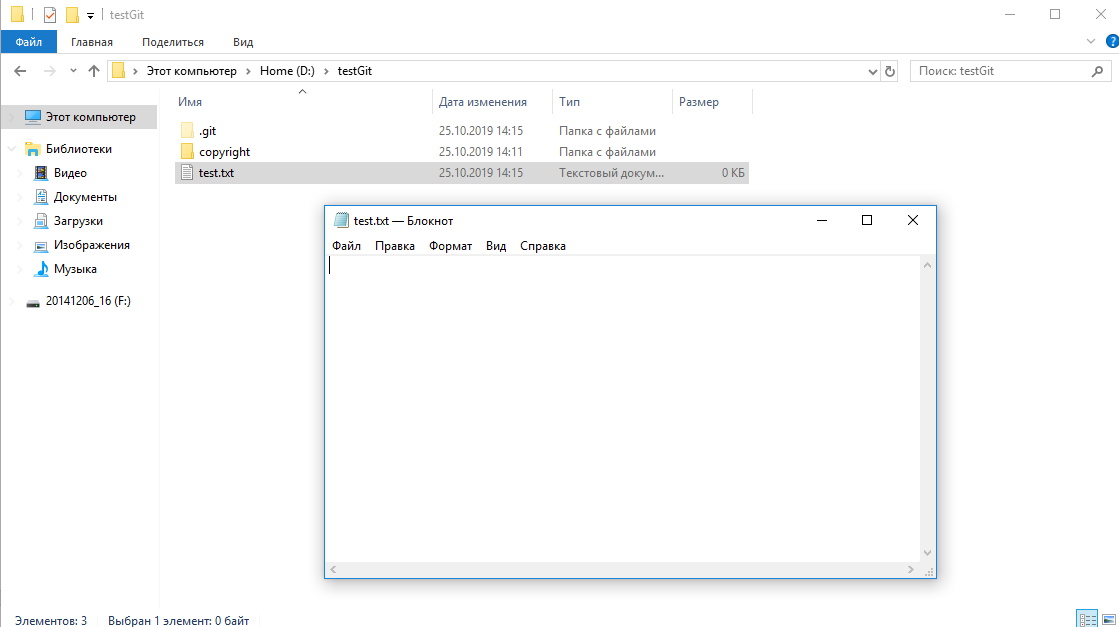
Теперь перейдем к самому интересному, вернемся к нашему репозиторию

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**

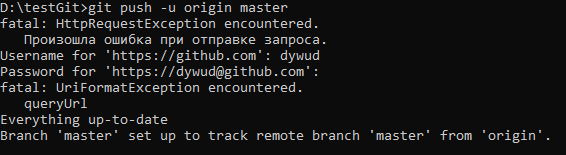


Как вы помните, первый коммит у меня был с пустым текстовым файлом, если зайти в папку, то увидим содержимое папки при первом сохранении



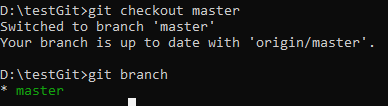
Таким образом вы можете переходить между своими коммитами и отменять какие-либо изменения.

Если при очередном сохранении вылезла ошибка вида:

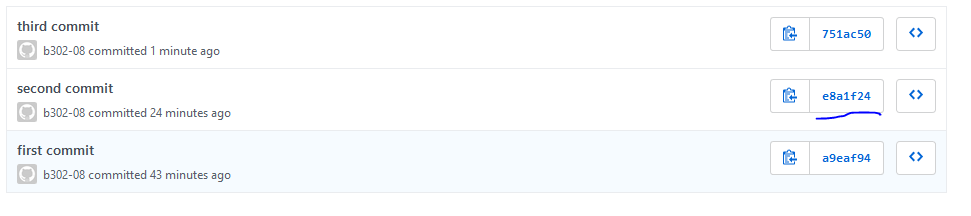


Это означает, что вы находитесь не на ветке мастер, ветки можно проверить командой **git branch**, \* означает то, на какой ветке вы щас находитесь, если эта звездочка стоит не возле ветки **master**, тогда нужно прописать команду **git checkout master**

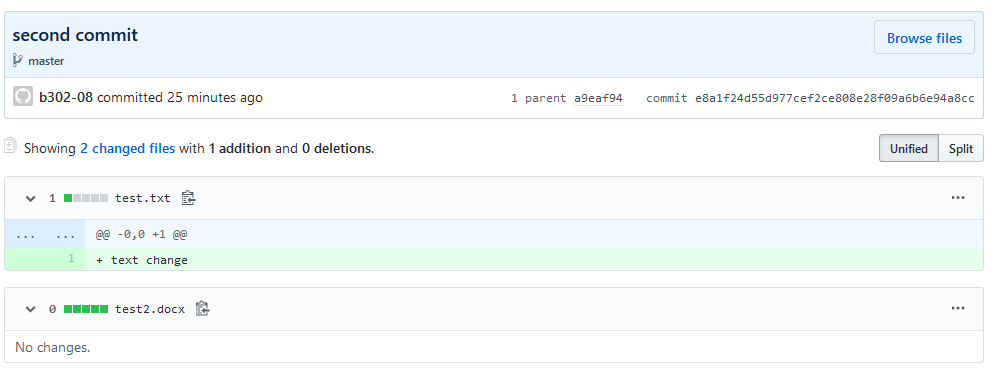




Если вы в GitHub перейдете в свои коммиты и нажмете на код коммита



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

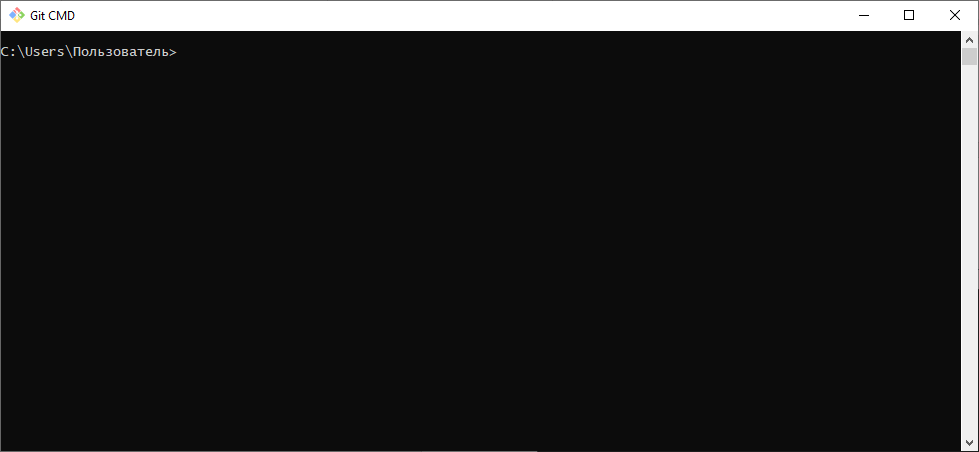


**Руководство по использованию gogs**

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:



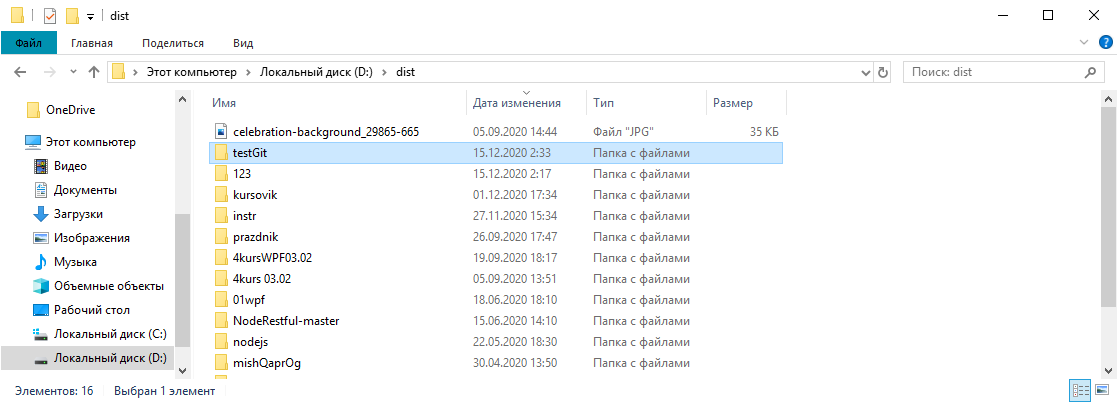
Далее нам необходимо перейти в нужную нам папку где мы будем создавать репозиторий. Переход осуществляется путем ввода в git команду **cd путь:**, для перехода на уровень ниже, необходимо написать **cd ..** для перехода на другой диск можно просто написать **d:**



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



После этого на диске D: появится папка.



Командой cd название\_папки мы переходим непосредственно в нашу папку



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.

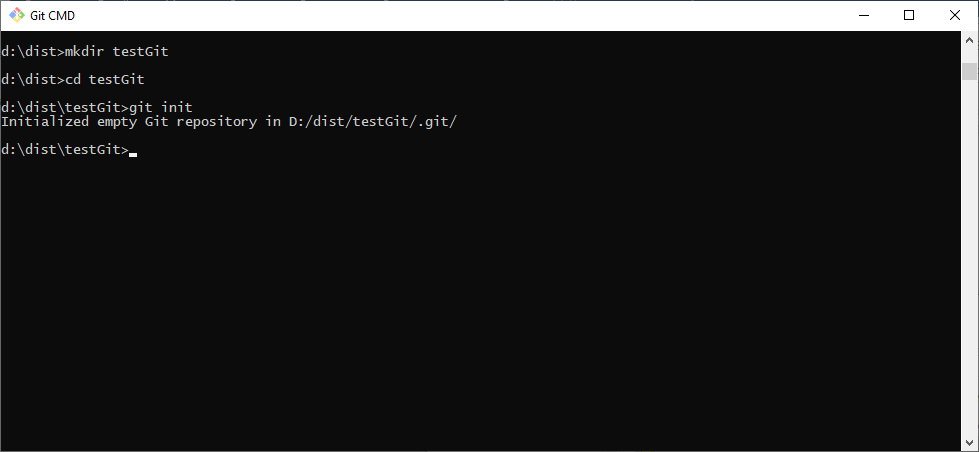
После того как мы оказались в нужной нам папке, мы вводим следующую команду:

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

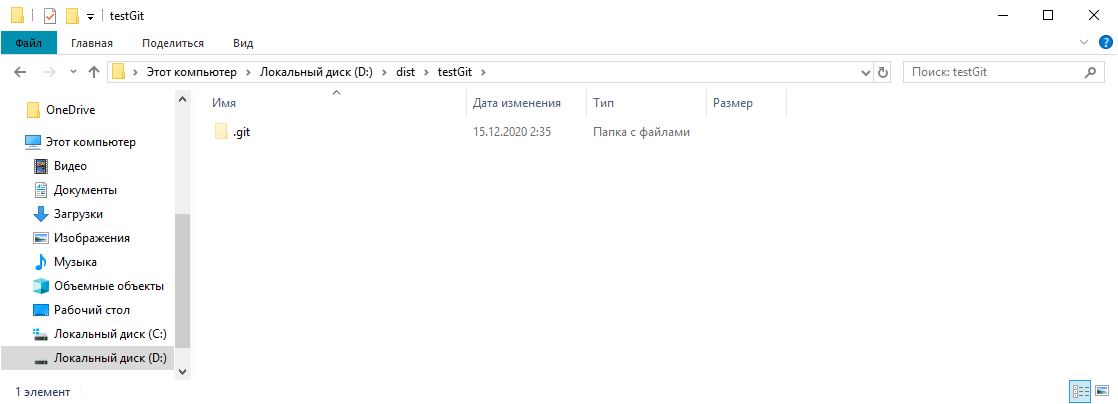
Она позволит нам работать с git в полном доступе (это необходимо делать только в колледже, потому что тут присутствует прокси, если вы занимаетесь этим дома, то эту операцию производить не нужно).

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

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



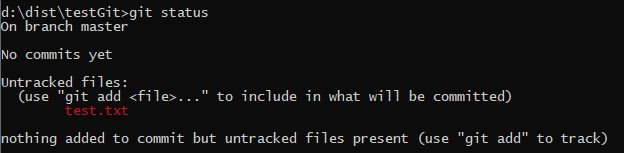
Если вы увидели надпись **Initialized empty Git repository in …**, значит репозиторий успешно создан в вашей папке. В папке появилась скрытая папка под названием **.git**, если же у вас нет этой папки, тогда необходимо нажать **Вид – Параметры – Изменить параметры папок и поиска**, затем перейти во вкладку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

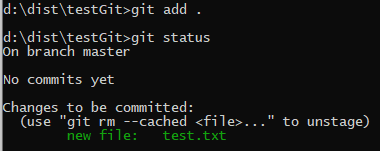
Если мы добавим в (!)нашу папку (не в папку .git) текстовый документ, то получим следующие:



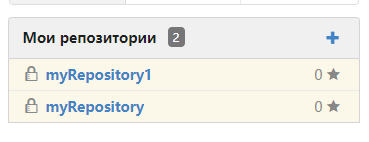
Красным подсвечиваются файлы, которые были добавлены, либо изменены.

Далее нам необходимо внести эти изменения в наш репозиторий командой

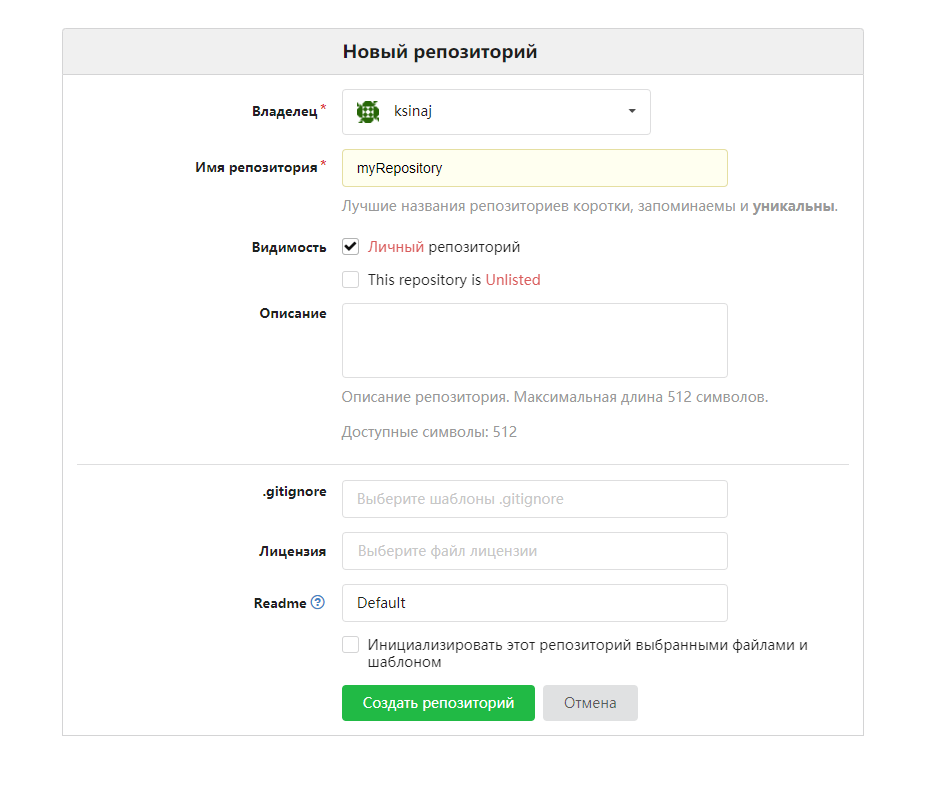
**git add . –** добавляет все файлы, содержащиеся в папке в наш репозиторий, точка после add означает, что будут добавлены все файлы, вместо точки можно добавлять конкретные файлы. После того как мы прописали команду **git add .**, далее мы пишем **git status** и там показывается, что были добавлены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



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 плюсик



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



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:

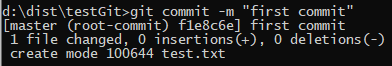
**git remote add origin** [**https://try.gogs.io/ksinaj/testGit.git**](https://try.gogs.io/ksinaj/testGit.git)

**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”**



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):

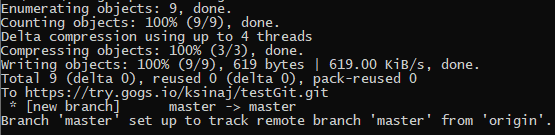
**git remote add origin https://github.com/ksinaj/test.git**

далее:

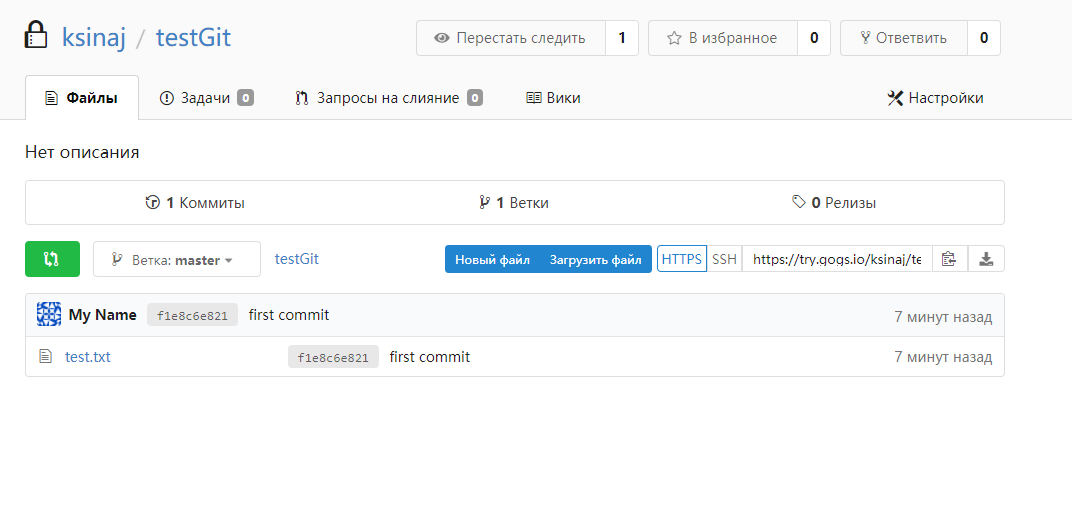
**git push -u origin master**

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



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)



**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 рассекречивать 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.