В данной методичке объясняется как правильно пользоваться средствами Git максимально понятно и просто, без лишних действий для минимальной работы с удаленными проектами.

Для начала разберем, что такое Git и перестанем путать его с Github.

Git – система контроля версий, которая позволяет отслеживать и записывать любые изменения в наборе файлов и возвращаться к предыдущим версиям проекта.

Github – веб-сервис для хостинга IT-проектом и их совместной разработки, основанный на системе контроля версий Git.

А теперь простыми и грубыми словами, чтобы каждый понял смысл и разницу между ними.

Git – это программа, которая добавляет специальные консольные команды для системы контроля версий. То-есть сохранение версий проекта, с последующим перемещением между ними.

Github – это веб-сайт, который даёт возможность сохранять к себе удаленные проекты. То-есть это облачный хостинг, такой же как Яндекс.Диск, Гугл.Диск, Gogs и другие (Из-за похожих названий Git и Github новички часто путают эти вещи).

Если всё еще не понятно что такое Git и для чего он нужен, привожу небольшой пример:

Большинство играли в компьютерные игры, возьмем любую сюжетную игру (например: GTA 5), представим, что мы прошли 5 первых миссий, каждый раз после которых у нас сохраняется прогресс, чтобы после перезапуске игры не проходить всё заново, но вдруг мы решили что прошли 2 миссию не верно (не выполнили достижение, не собрали нужный предмет на будущее или что-то еще), тогда с помощью своих сохранений вы можете откатить весь свой прогресс на предыдущие сохранение. Как раз это и называют - Система Контроля Версия (СКВ), которую позволяет проводить программа Git.

Теперь попробуем использовать Git, для начала его нужно установить. Перед этим можно проверить не устанавливался ли на данном компьютере ранее.

Для этого откройте любой терминал, к примеру cmd, и напишите в командную строку слово «git», если ответом вы получите данное сообщение (Рисунок 1.):

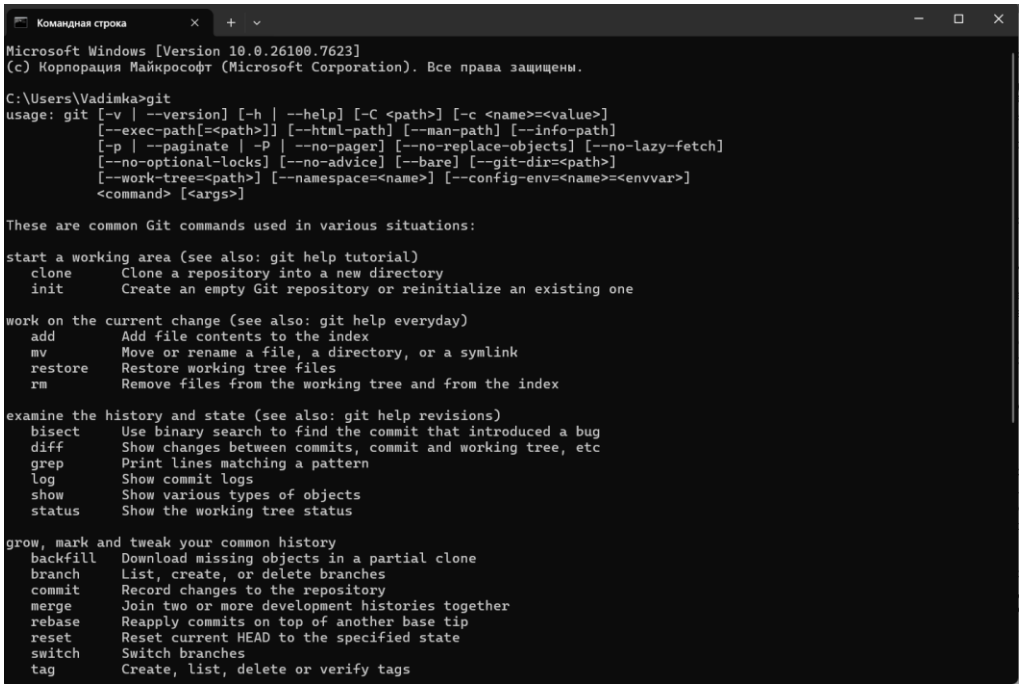

Рисунок 1.

значит Git установлен на вашем ПК, в противном случае вы получите другое сообщение (Рисунок 2.):

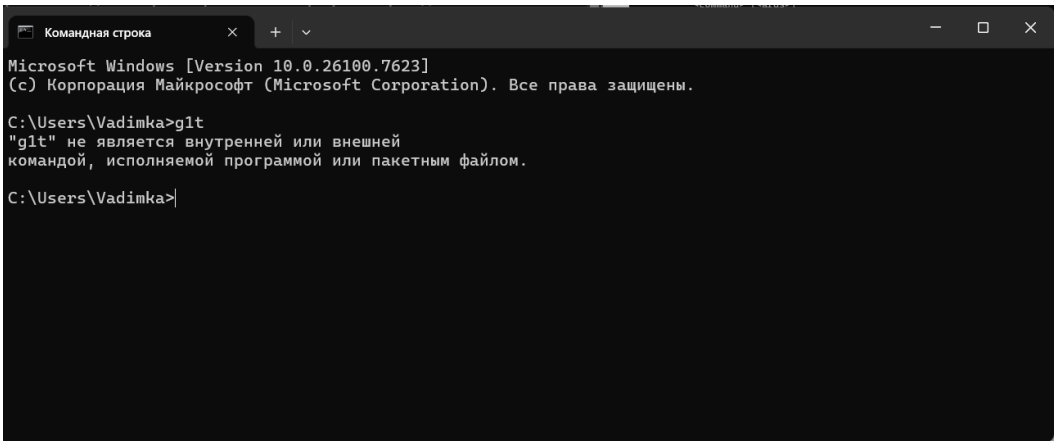

Рисунок 2.

Так, как у меня программа уже установлена, то для примера я специально написал команду с ошибкой, но вам нужно писать именно «git».

Как только убедитесь что у нас нет установленной программы, переходим на официальный сайт https://git-scm.com/ , на котором нажимаем кнопку «Install for Window» (Рисунок 3.)

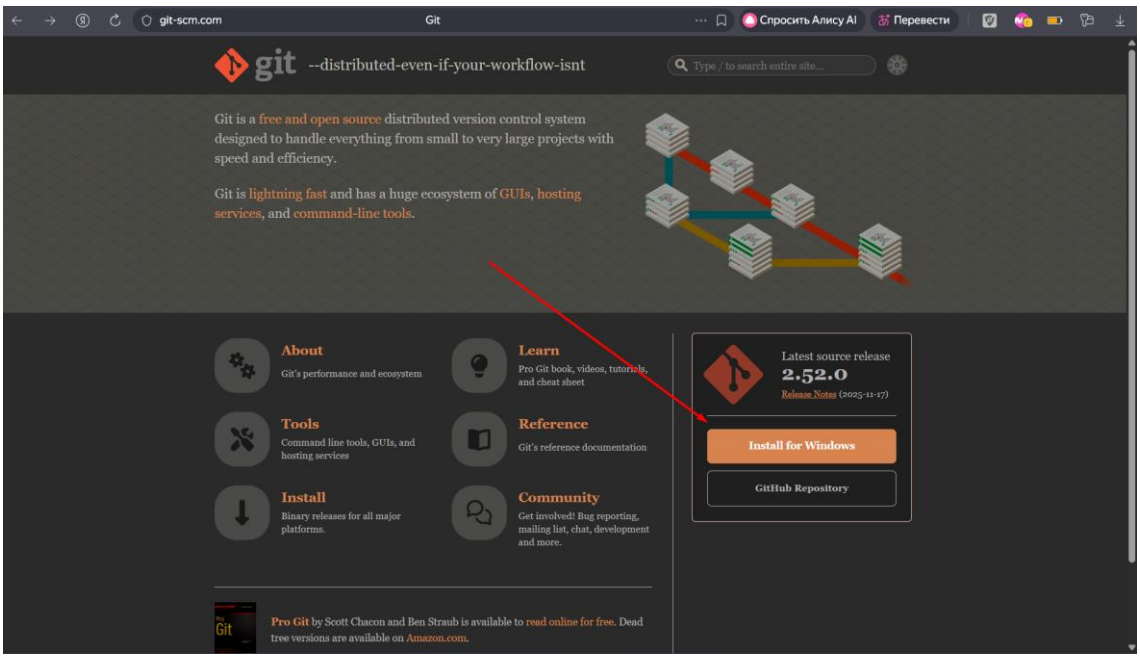

Рисунок 3.

на следующей вкладке выбираем «Windows» (если у вас другая операционная система, выбираем соответствующую), выбираем пакет «Standalone Installer» (Это последняя стандартная и стабильная версия), нажимаем «Git for Windows/x64 Setup» (Рисунок 4.).

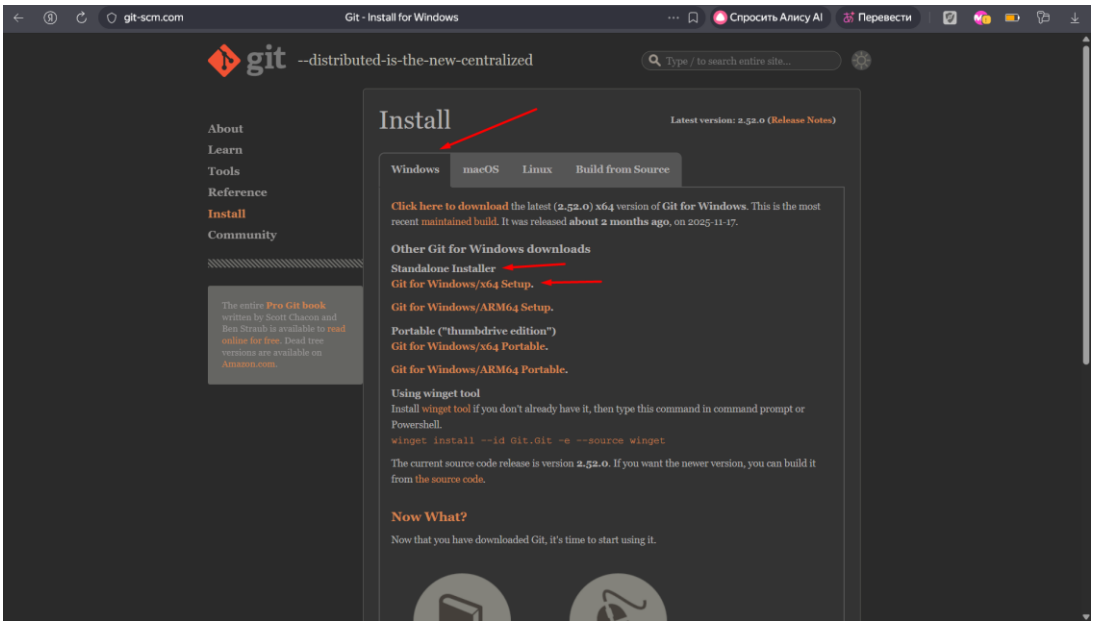

Рисунок 4.

У нас скачается установочный .exe файл (Рисунок 5.), его нужно запустить и пройти стандартную установку. Внутри автоматически подбираются стандартные настройки, так что если не разбираетесь, то лучше оставить всё как есть и нажимать «Next» до окончания установки.

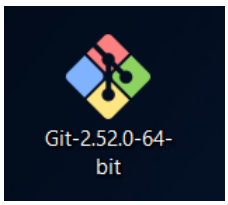

Рисунок 5.

После успешной установки мы можем еще раз проверить точно ли установился Git на компьютер, снова открываем любой терминал и вводим команду «git», там мы должны увидеть ответ как на Рисунке 1.

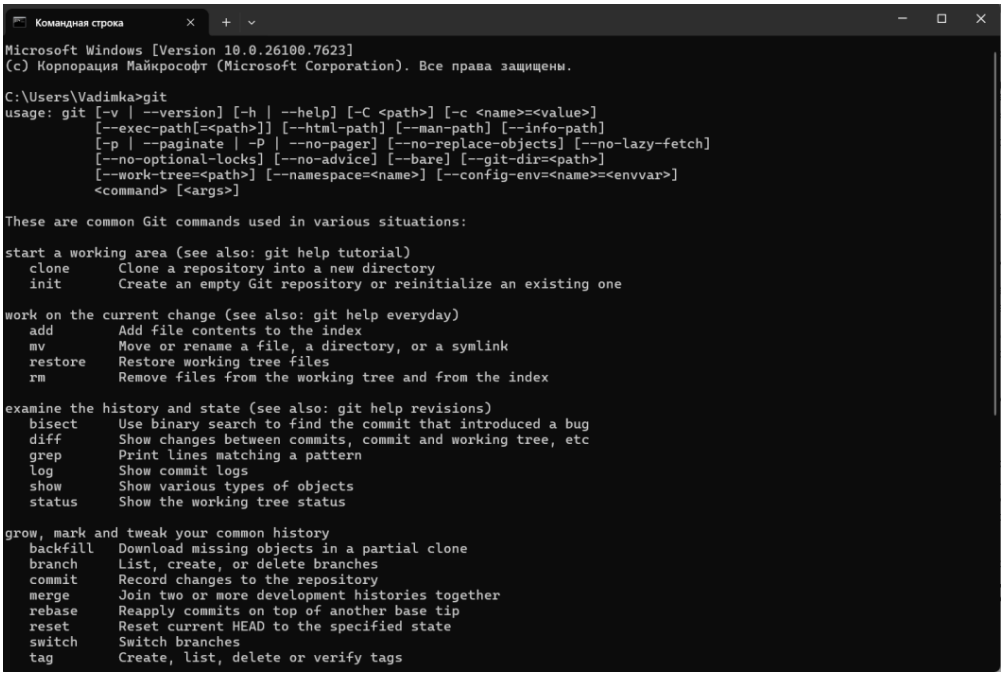

Рисунок 1.

В ответе вам кратко обозначают все возможности и доступные команды для работы с Git.

Важно подметить: Git грубо говоря, добавляет на ваш ПК команды, а пользоваться этими командами можно в любом терминале, такими как стандартными от операционной системы (Например: Командная строка/cmd или Windows PowerShell (Рисунки 6, 7)) и новым терминалом, который устанавливает сам Git, называется «Git Bash» (Рисунок 8.). Пользоваться можно чем вам угодно, зависит от личных предпочтений, но в рамках этого урока мы будем работать с данной нам программой Git Bash.

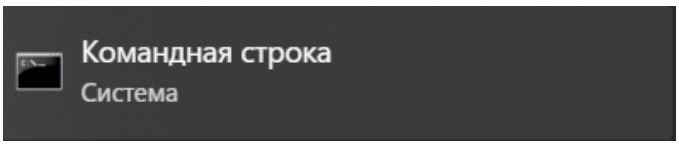

Рисунок 6.

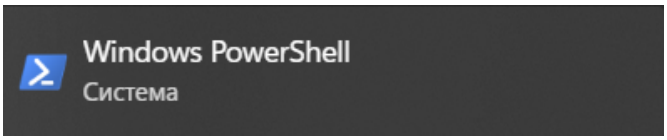

Рисунок 7.

Рисунок 8.

Теперь, мы наконец-то можем перейти к использованию Git. Чтобы обращаться именно к этой программе нам каждый раз перед своим запросом нужно писать волшебное слово «git», а после него используемую команду. Но перед тем, как что-то делать, нам нужно задать некоторые настройки.

Дело в том, что каждую операцию, которую мы проводим записывается в системе, и перед тем как мы будем что-то делать, нужно подписать себя в системе. Для этого вводим следующую команду (Рисунок 9.):

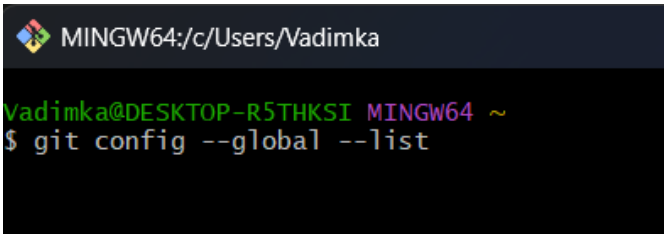

Рисунок 9.

Она покажет нам наши настройки, нас интересует только две строки «user.name» и «user.email», они должны быть заполнены. Для этого нам нужно написать следующие две команды (Рисунок 10.):

```
Vadimka@DESKTOP-R5THKSI MINGW64 ~
$ git config --global user.name "vadimtaop"

Vadimka@DESKTOP-R5THKSI MINGW64 ~
$ git config --global user.email "vadimmuhin77@gmail.com"

Vadimka@DESKTOP-R5THKSI MINGW64 ~
$ git config --global --list
user.name=vadimtaop
user.email=vadimmuhin77@gmail.com

Vadimka@DESKTOP-R5THKSI MINGW64 ~
$
```

Рисунок 10.

в кавычках нужно указать своё имя и почту, после можно снова ввести команду с проверкой наших настроек (Рисунок 9.) и убедиться, что всё наши две строки заполнены.

Данная настройка проводиться только 1 раз после установки Git, в дальнейшем ее использование не нужно, ведь настройки уже заданы.

Следующим этапом нужно подготовить GitHub, чтобы работать удаленно с нашими проектами. Для этого переходим на официальный сайт https://github.com/ и проходим регистрацию (Рисунки 11, 12) если вы зашли впервые или входим, если уже были ранее.

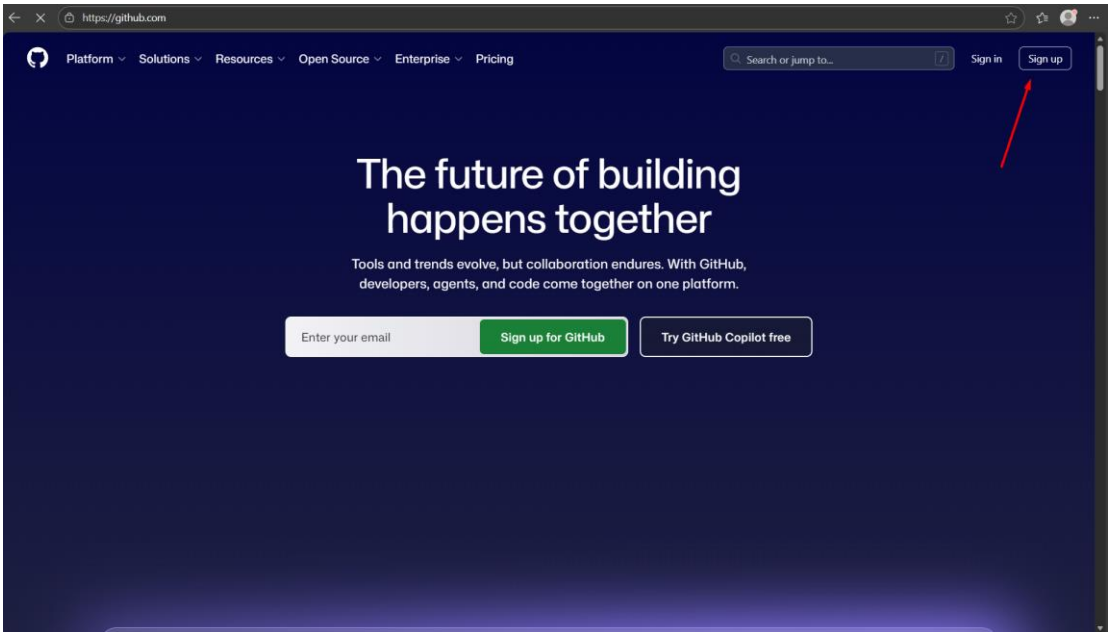

Рисунок 11.

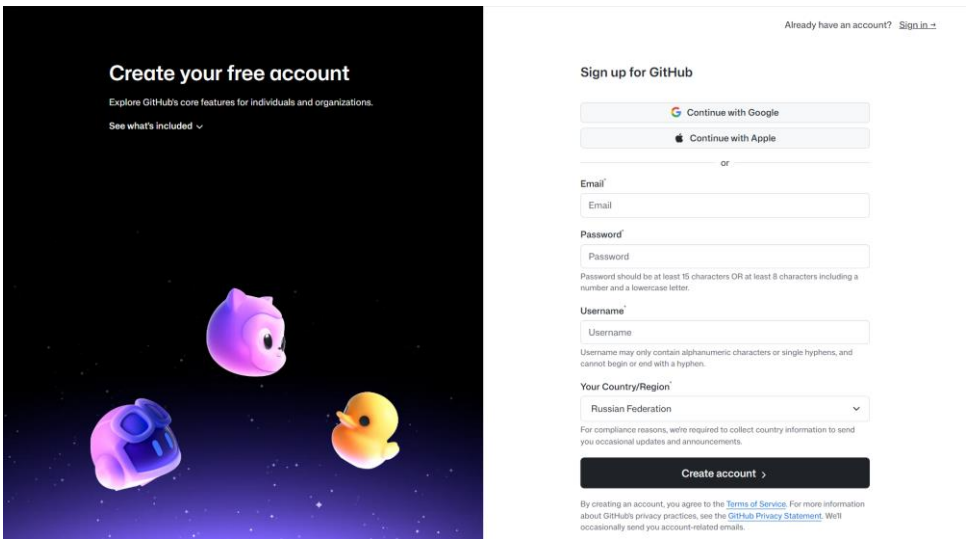

Рисунок 12.

После авторизации мы попадаем на главное меню (Рисунок 13.), в ней есть много информации, но в рамках нашего обучения сейчас нам понадобиться только панель слева, здесь находятся наши удаленные репозитории (В программировании проектами называют репозиториями), нажимаем кнопку «New», для создания нового, в анкете (Рисунок 14.) прописываем название, выбираем публичный репозиторий (или приватный если хотим скрыть его от других пользователей), остальное пока что можно оставить. Нажимаем кнопку «Create repository»

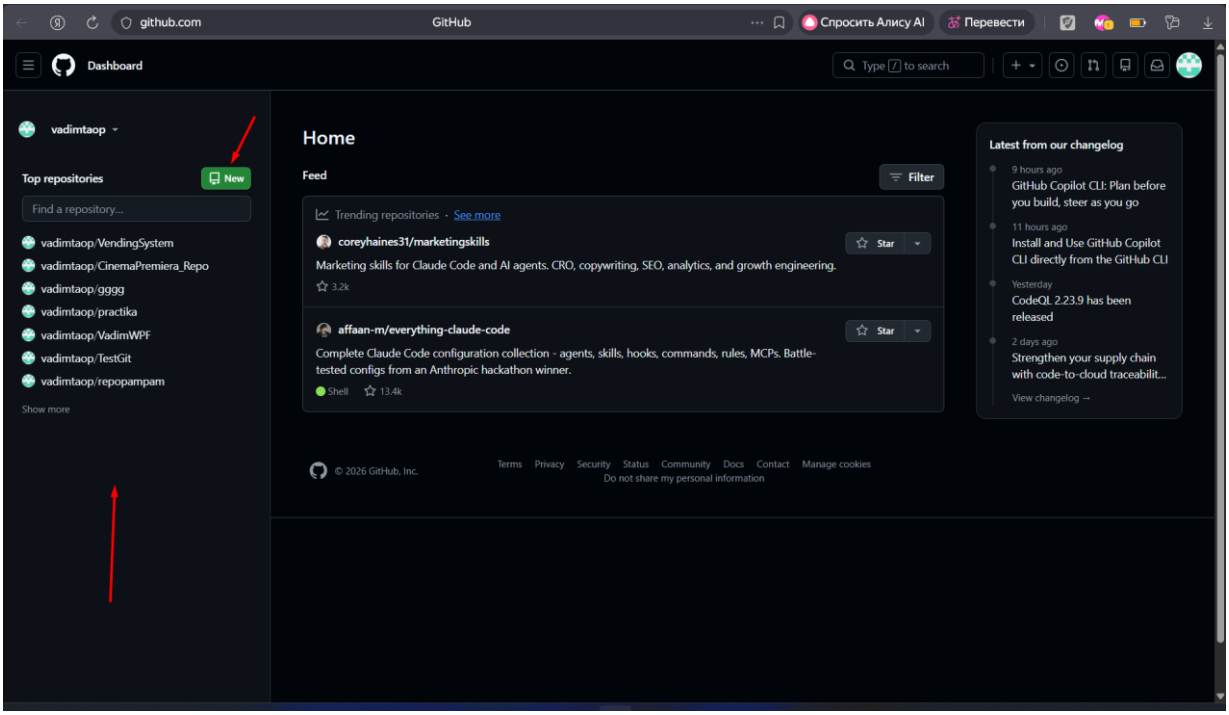

Рисунок 13.

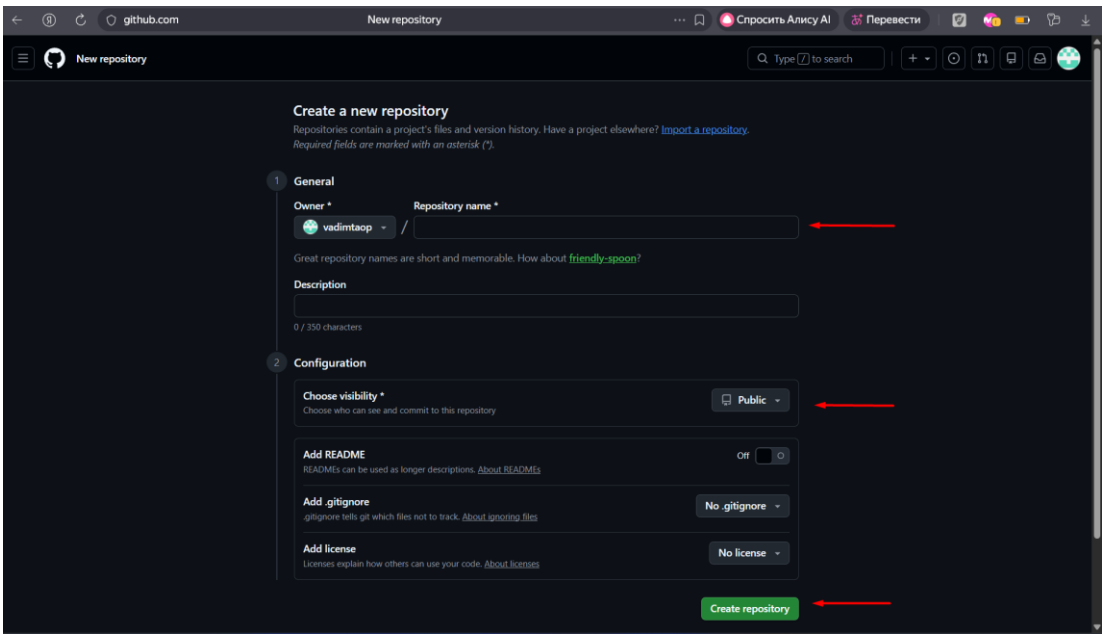

Рисунок 14.

После создания у вас появиться подобное окно с краткой инструкцией как подключить удаленный репозиторий (Рисунок 15.)

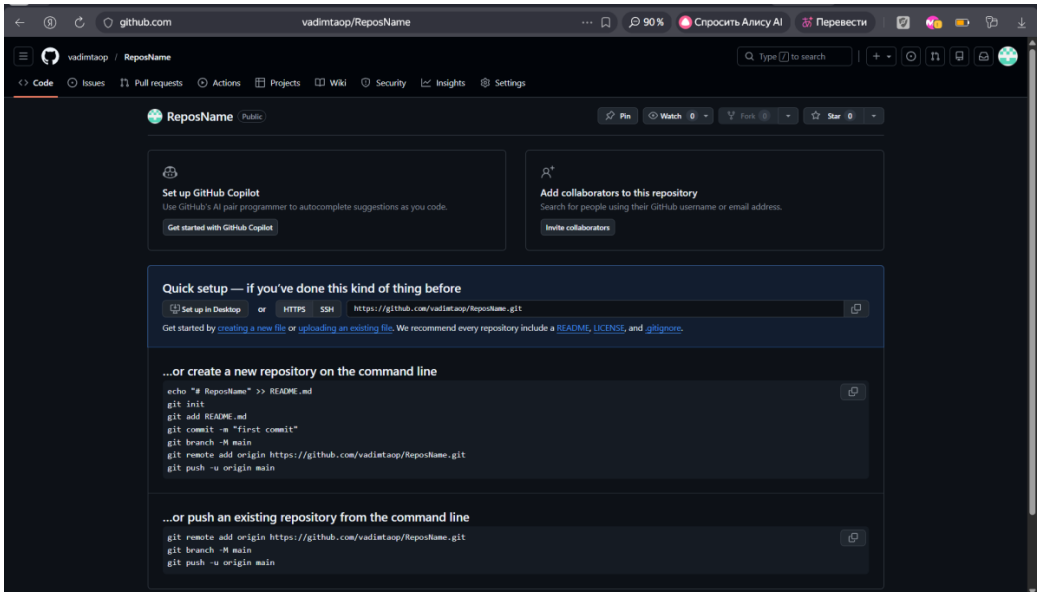

Рисунок 15.

Возвращаемся обратно на наш компьютер. Создадим папку на рабочем столе и представим, что это наш проект, в ней для примера добавим текстовый блокнот (Рисунок 16.).

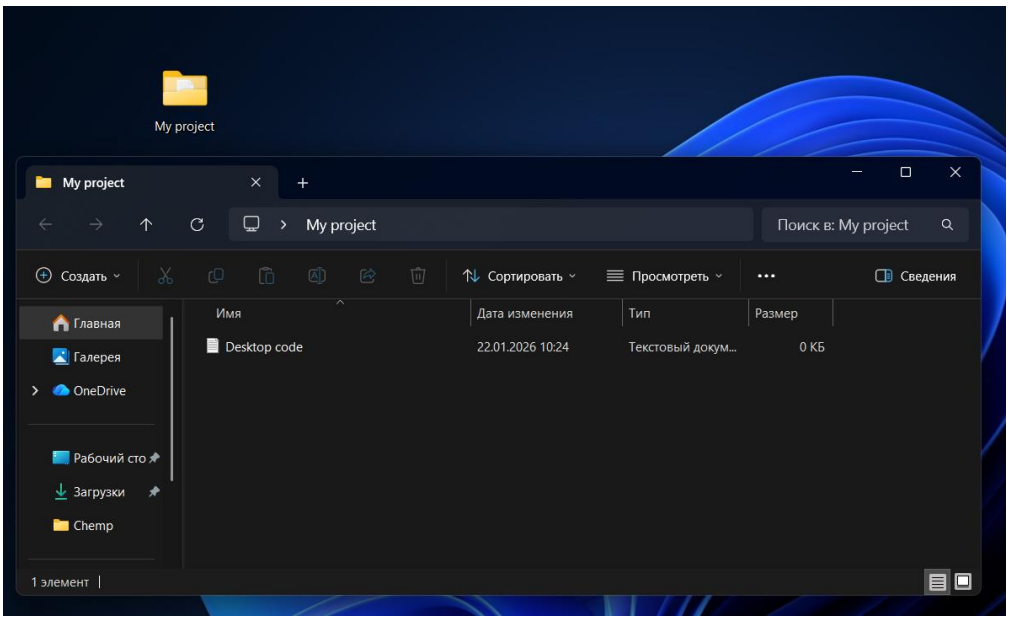

Рисунок 16.

Теперь открываем терминал (Git Bash), чтобы сохранить наш проект. Для начала нам нужно перейти в эту папку, по умолчанию в терминале мы находимся в системной папке пользователя (C:\Users\Vadimka), если написать стандартную команду из windows, которая пишется так: «ls» и выводит нам список файлов, которые находятся в этой папке. Можем в этом убедиться (Рисунок 17.).

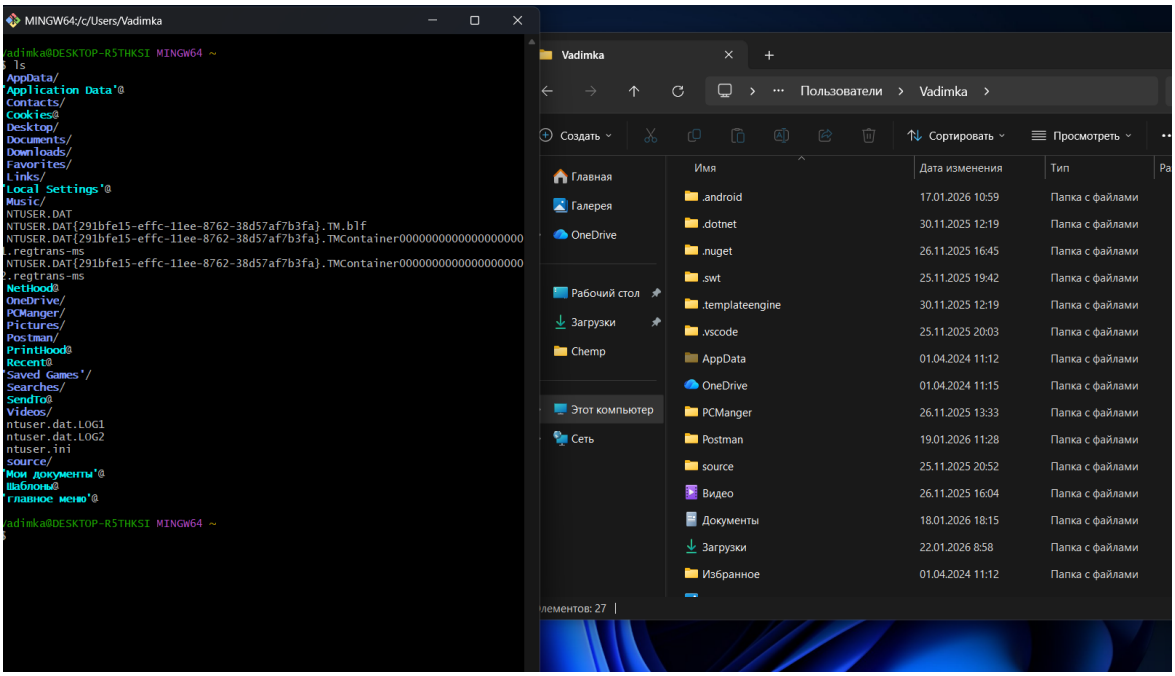

Рисунок 17.

Чтобы перемещаться по папкам, нужно использовать команду «cd путь». Если написать просто «cd», то нас перебросить в первоначальную папку. Так как папка у нас находиться на рабочем столе пишем следующее (Рисунок 18.):

Рисунок 18.

Важно подметить, что если в пути присутствуют пробелы или русские буквы, то его нужно писать в кавычках. Теперь мы находимся в папке My project. Чтобы узнать в каком пути мы находимся, нужно смотреть на разноцветную строку (Рисунок 19.), которая пишется каждый раз перед новым запросом. Чтобы переместиться на папку назад, можно ввести две точки, команда будет выглядеть так «cd ..».

Сейчас я переходил по одной папке, но можно скопировать путь папки и прописать его полностью (Рисунок 19.):

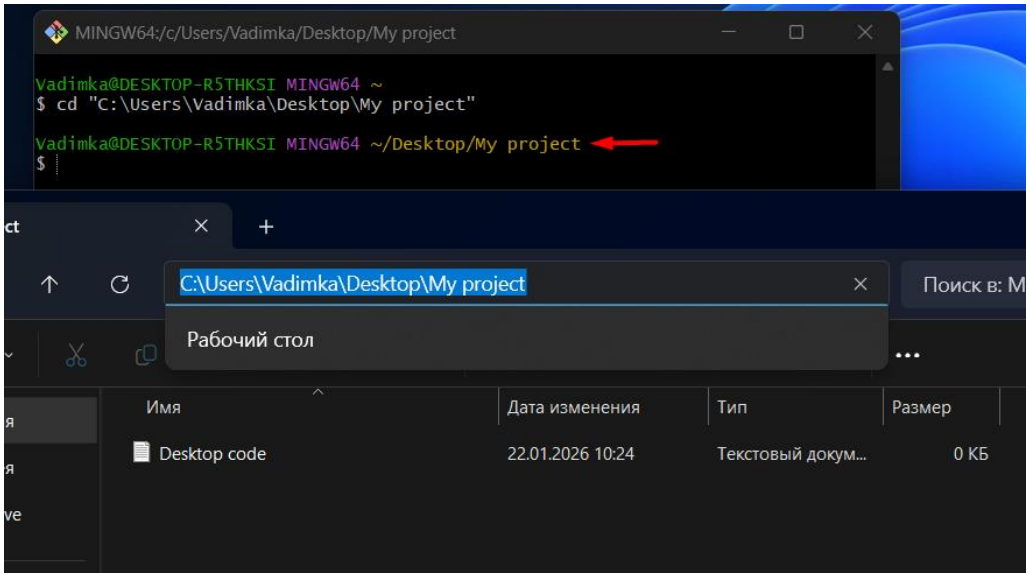

Рисунок 19.

Важно подметить, что если вам надобиться перейти в системную папку (например: C:\Windows\System32), то терминал нужно запускать от имени администратора, иначе ответом будет ошибка о недостатке прав.

Так-же чтобы не мучаться с переходом на пути, при установки Git, добавляется

функция, с помощью которой можно открыть нужную папку сразу через контекстное меню, для этого нажмите ПКМ по нужной папке, и выберите пункт «Open Git Bash here» (Рисунок 20.):

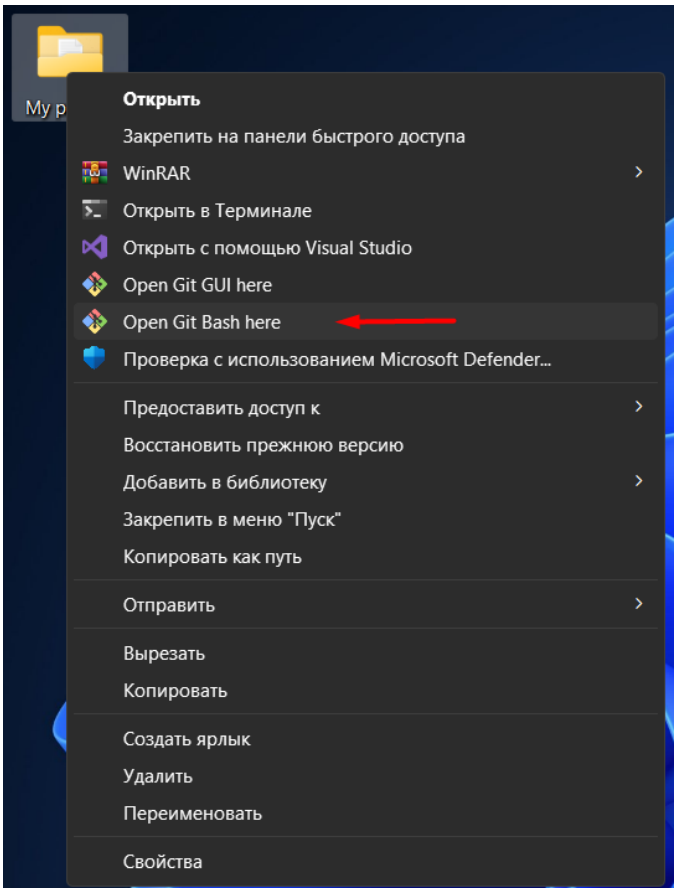

Рисунок 20.

После того как мы попали в нашу папку через терминал, нужно внедрить Git в наш проект, для этого нужно написать команду «git init» (Рисунок 21.). Ответом поступит фраза о том, что Git установлен. Так-же появиться скрытая папка .git (если в операционной системе у вас отключена функция видеть скрытые папки, то вы не сможете ее увидеть), а еще в той самой разноцветной строке после пути появится надпись «master», она указывает на название ветки, в которой вы находитесь, по умолчанию создается ветка «master».

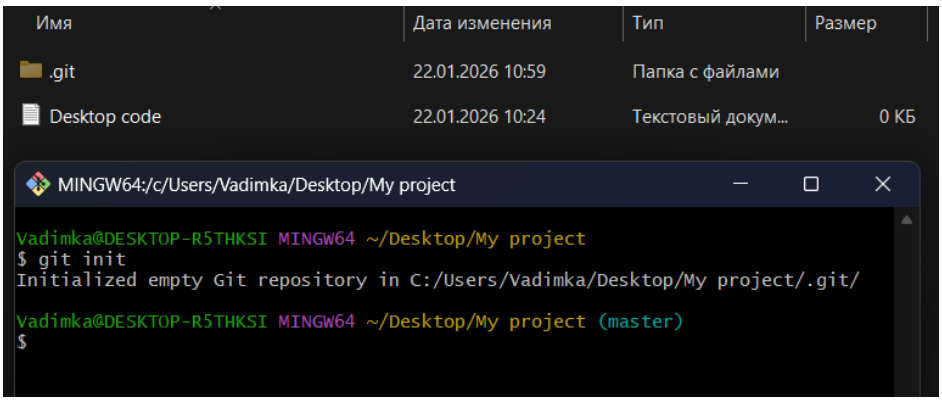

Рисунок 21.

Следующим этапом, нам нужно выбрать файлы, которые мы хотим сохранить, для этого используем команду «git add Название_файла», чтобы выбрать все файлы, находящиеся в папке, можно написать точку или звезду (Рисунок 22.):

Рисунок 22.

Теперь нам нужно обязательно закомментировать наши выбранные файлы. Для этого используется команда «git commit -m “комментарий”» (Рисунок 23.).

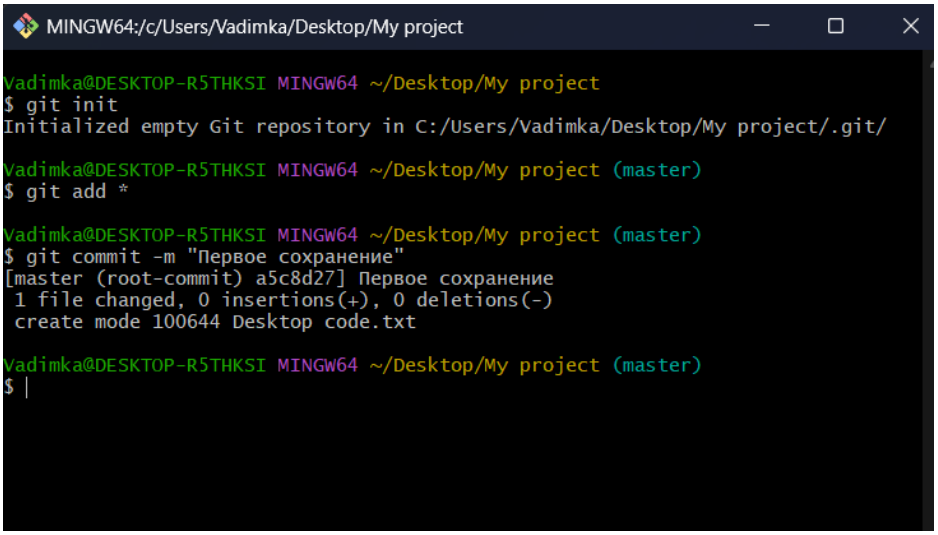

Рисунок 23.

Теперь наши файлы сохранены локально, и в будущем мы можем

переключаться между ними. Попробуем выгрузить их на удаленный хостинг, то-есть на Github. Для это нам сначала нужно подключить к нашему ранее созданному репозиторию, обратим внимание на Рисунок 15.

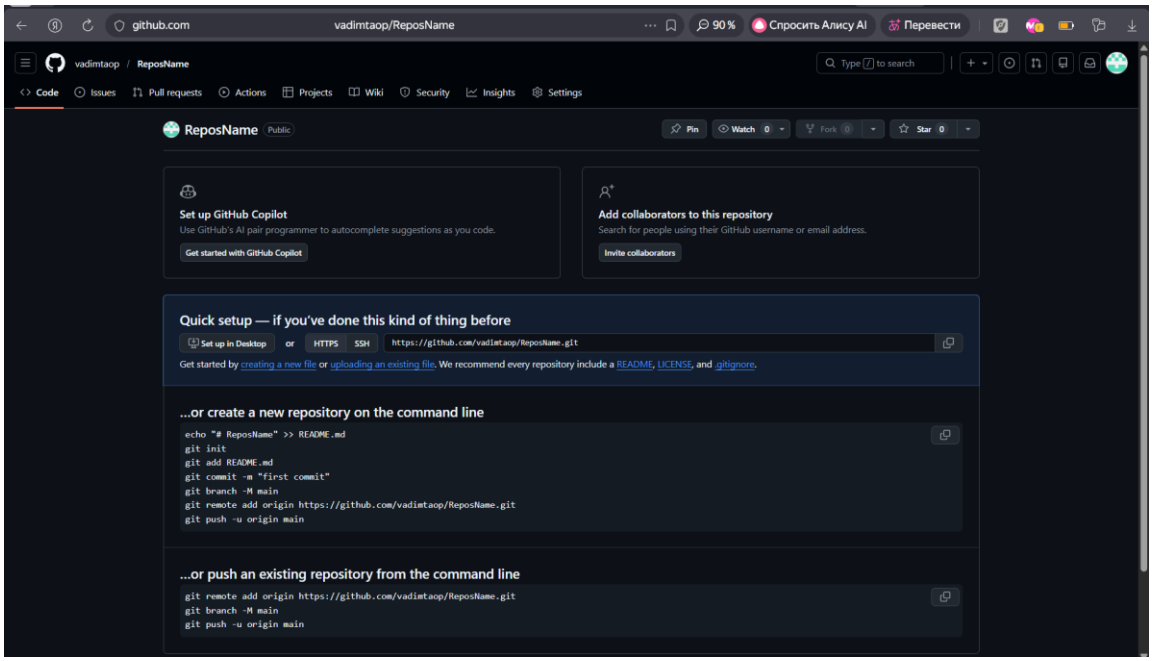

Рисунок 15.

В параметрах проекта находится ссылка на репозиторий двух видов, HTTPS и SSH, нам нужна только HTTPS. Копируем ссылку и возвращаемся в терминал, в котором нужно написать следующую команду, «git remote add origin ссылка» (Рисунок 24.) (обратите внимание, что привычные ctrl+c, ctrl+v в терминале отвечают за другие действия, для копирования и вставки нажми ПКМ и выберите пункты «Copy» и «Paste»):

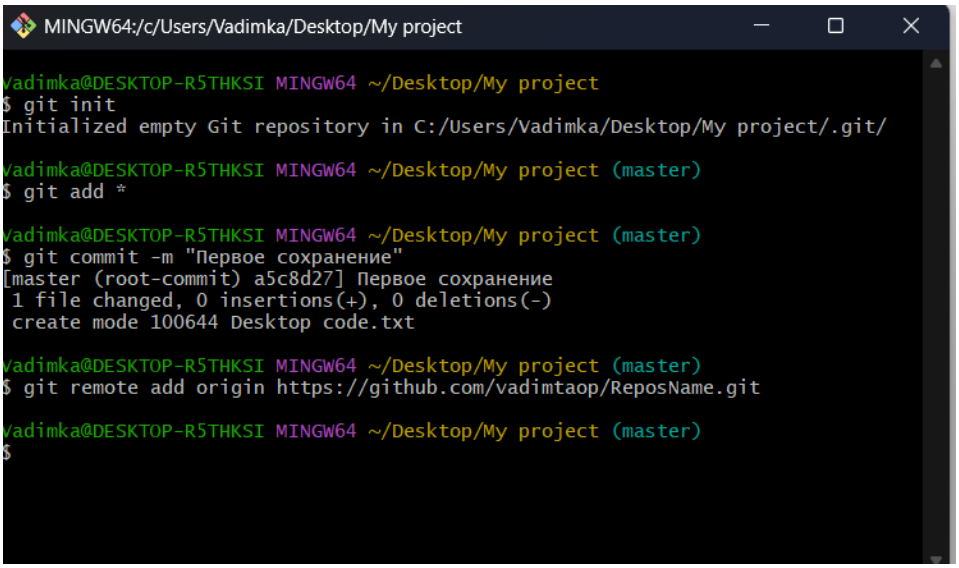

Рисунок 24.

Если после попытки подключения к удаленному репозиторию ответом выводится ошибка, то проверьте подключение к интернету, включите или выключите прокси-сервер/VPN.

Репозиторий нужно подключать только один раз, при дальнейших выгрузках нужно лишь добавить новые файлы («git add»), закомментировать («git commit») и спокойно выгрузить.

Последним этапом мы выгружаем наш проект на удаленный репозиторий, для этого используется команда «git push». При первой выгрузки Git может выдать ошибку и потребовать указать какую ветку нужно выгрузить, поэтому ему нужно написать название ветки (в нашем случае это «master»), поэтому вводим команду «git push -u origin master» (Рисунок 25).

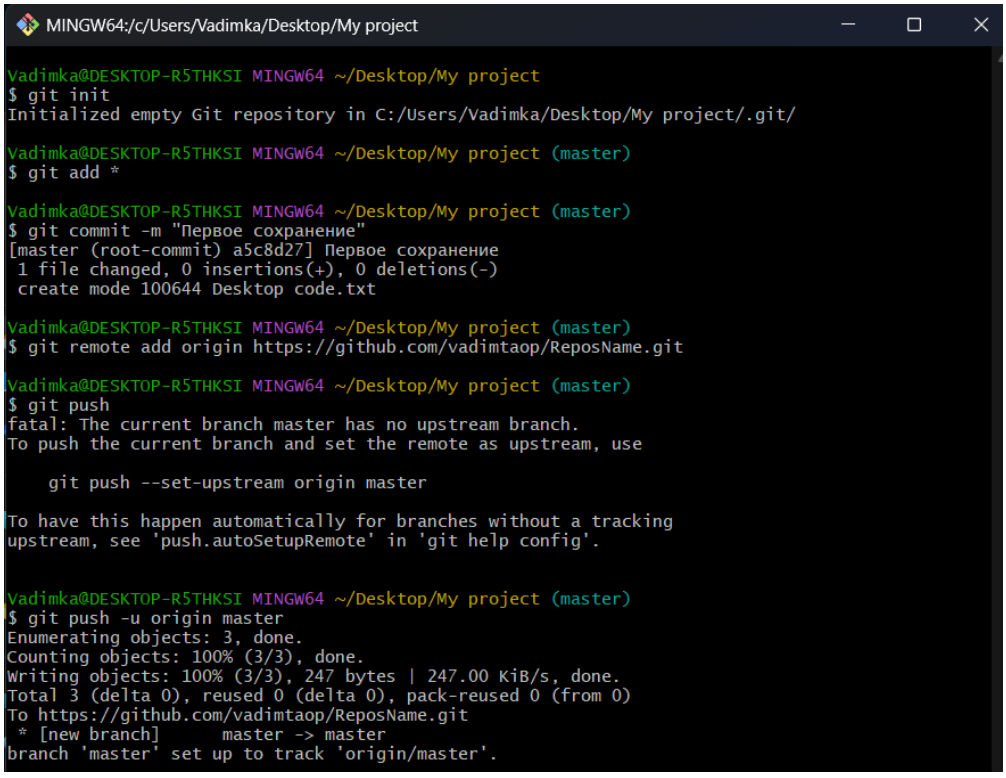

Рисунок 25.

Ответом поступит сообщение о том, сколько файлов было загружено на удаленный репозиторий и с какой скоростью, дождитесь когда пройдет полная загрузка. После перезагрузите страницу с вашим удаленным репозиторием на Github и убедитесь, что проект был загружен (Рисунок 26). Так-же там можно узнать много информации о проекте, например, такие как: название репозитория, комментарий на файлах, которые были изменены, отчет времени сколько назад

был выгружен тот или иной файл.

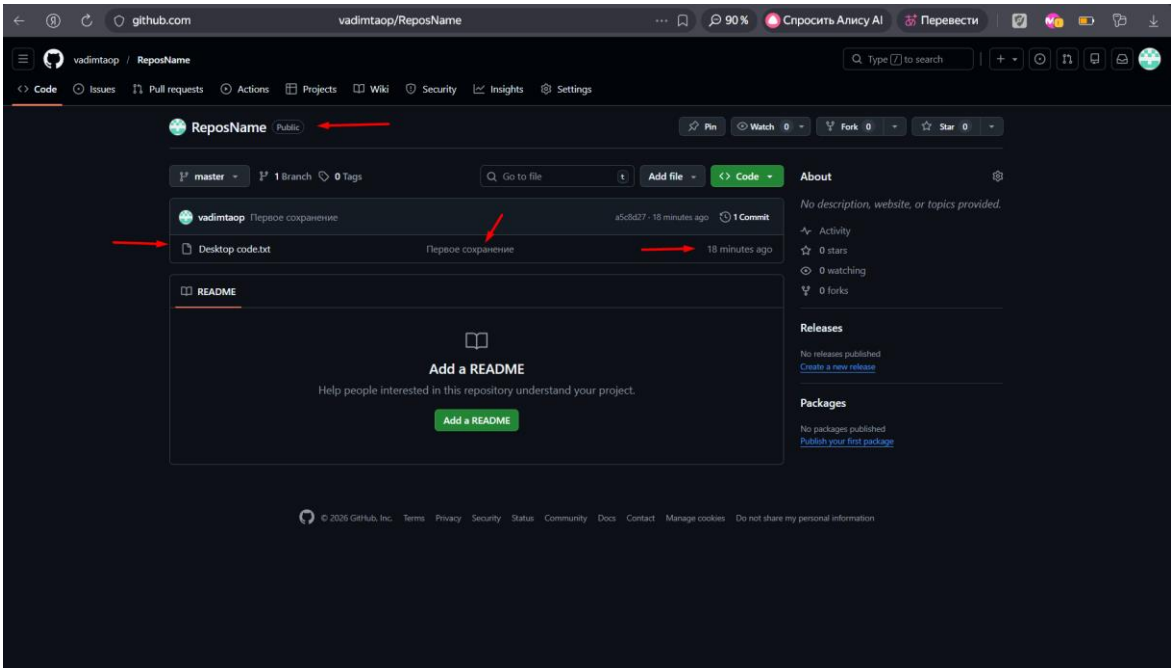

Рисунок 26.

Теперь напишем какой-нибудь текст в наш текстовый файл, и еще добавим новый (Рисунок 27.) для того, чтобы имитировать изменения в проекте.

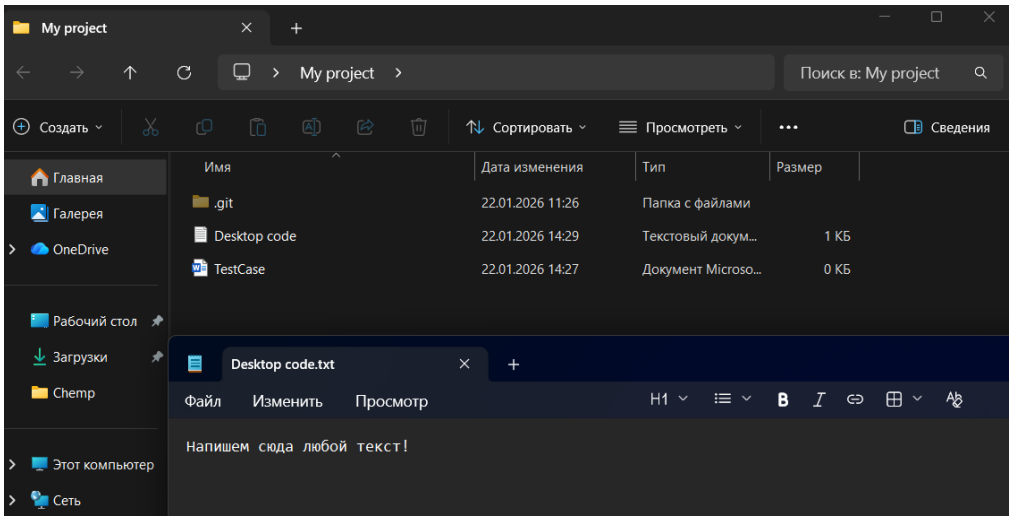

Рисунок 27.

Проект изменен. Теперь давайте обновим наш удаленный репозиторий на github. Для этого снова открываем терминал в данной папке. И выполняем команды, которые мы изучили ранее. Но выполнять нужно не все. Например, команду «git init» выполнять не нужно, ведь проект уже инициализирован, или «git remote», ведь проект так-же уже подключен к удаленному репозитории.

Чтобы сохранить изменения нам нужно написать следующие команды: «git add

*» для добавления всех файлов для обработки их git (файлы, которые не добавлены – не будут выгружено, проще говоря, git эти файлы не видит и не тронет), «git commit -m “комментарий”» для фиксирования изменений и указания комментария, а потом выгрузить с помощью команды «git push» (Рисунок 28.).

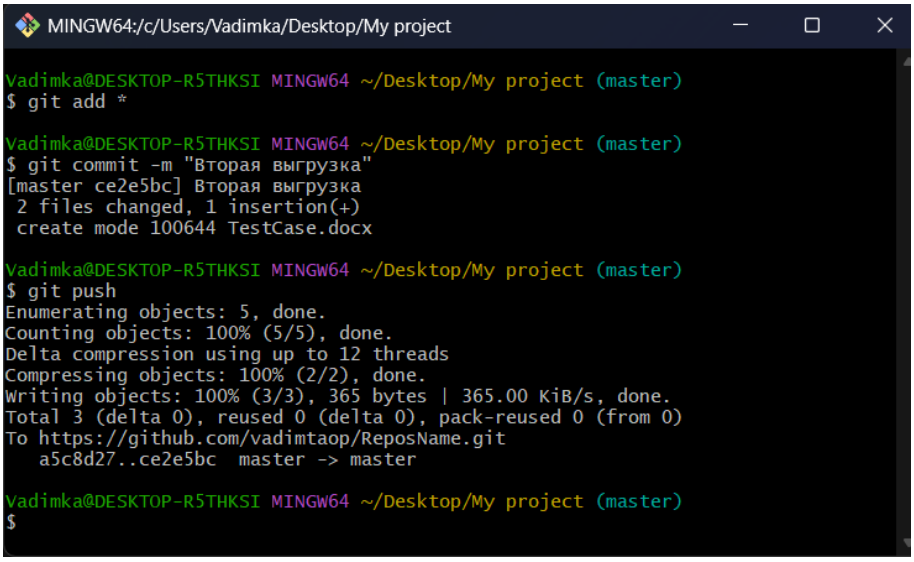

Рисунок 28.

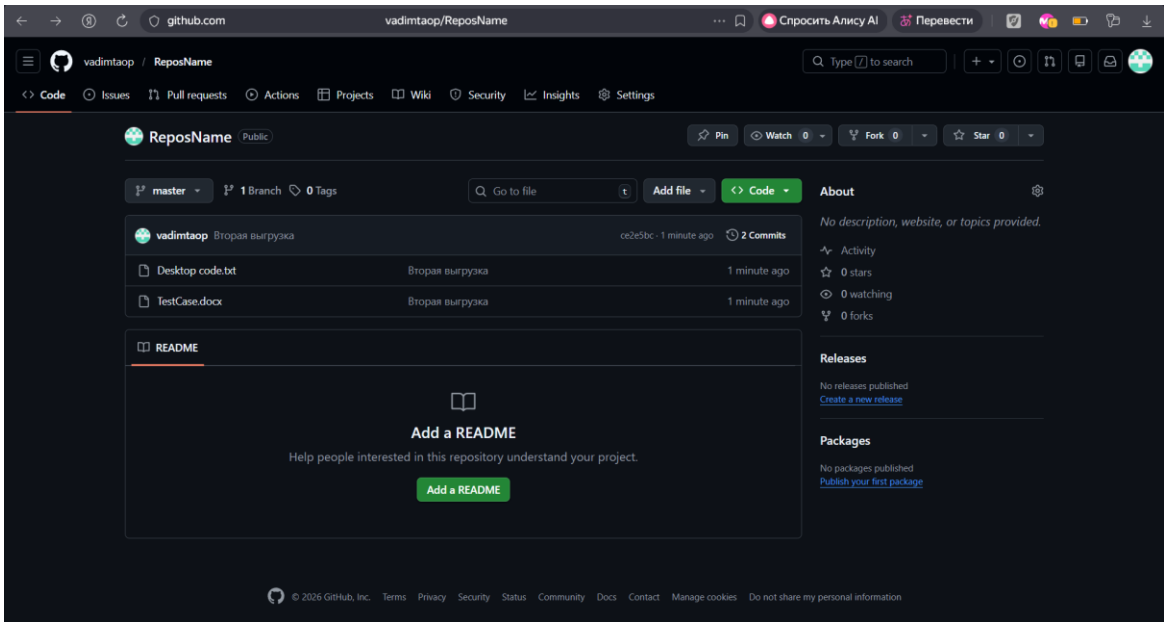

Рисунок 29.

После успешного выполнения команд, перезагрузите сайт на github и можете убедиться в том, что проект был выгружен на удаленный репозиторий. Теперь не трогая файл «TestCase», изменим текст в текстовом файле (Рисунок 30.).

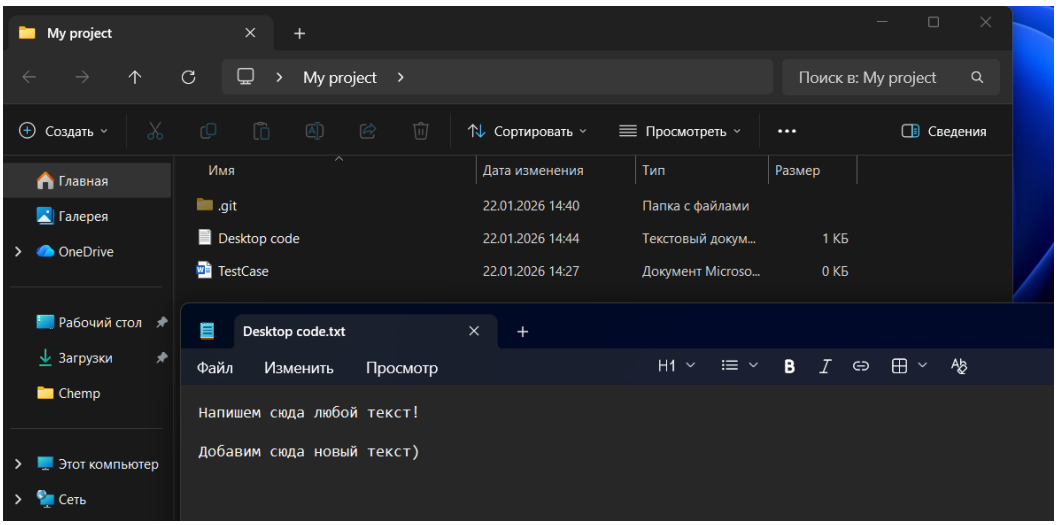

Рисунок 30.

Теперь проделываем всю ту же процедуру сохранения и выгрузки, но снова меняем комментарий на новый (Рисунок 31.).

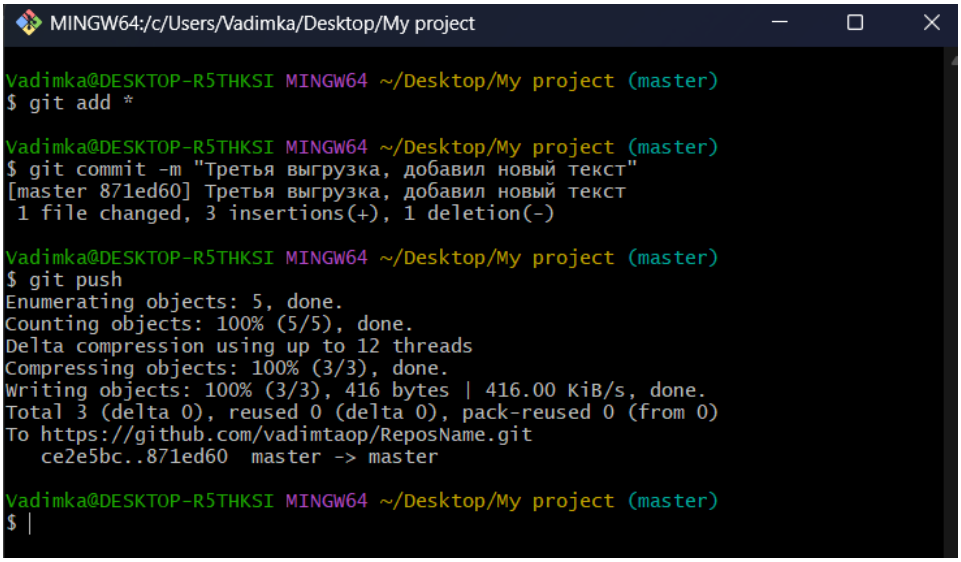

Рисунок 31.

Снова перезагружаем страницу на Github и можем заметить, что наш комментарий изменился только на текстовом файле, а в файле «TestCase» остался предыдущий (Рисунок 32.). Дело в том, что система Git сохраняет и выгружает только измененные файлы, например в отличии от того же Яндекс.Диск, где каждый раз нужно выгружать весь проект, что будет затрачивать очень много памяти. В этом и есть один из основных плюсов системы Git.

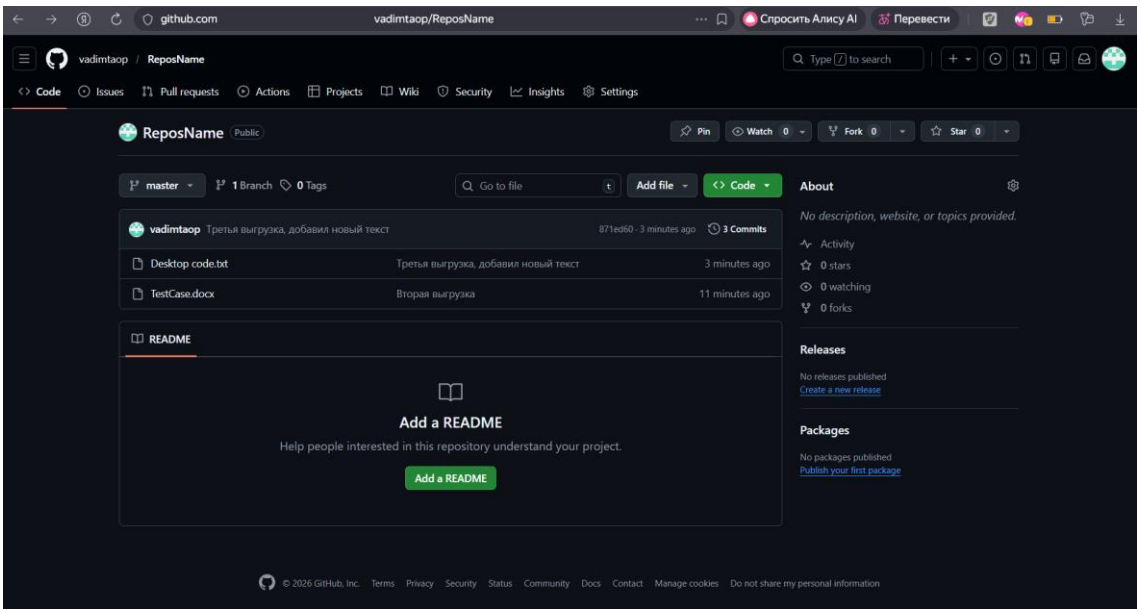

Рисунок 32.

Мы научились выгружать проект в удаленный репозиторий, теперь представим, что мы используем другой ПК или вообще другой пользователь. Попробуем скачать наш проект для дальнейшей работы с ним. Для этого создадим новую папку на рабочем столе, после нажнем по ней ПКМ и выберем пункт «Open Git Bash here» (Рисунок 33.).

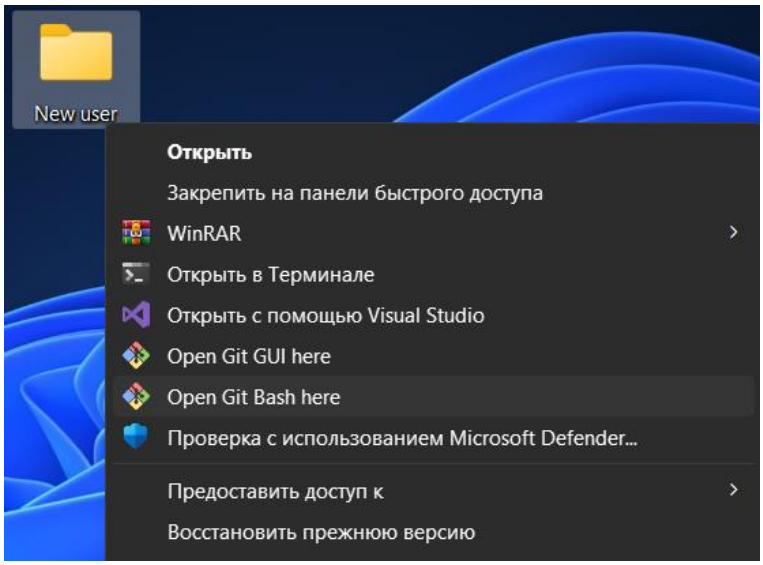

Рисунок 33.

Чтобы скачать удаленный репозиторий, нужно написать следующую команду: «git clone ссылка». Переходим на главную страницу с нужным репозиторием, нажмите на зеленую кнопку «Code», и скопируйте ссылку на репозиторий (Рисунок 34.), так-же в этом меню его можно сразу скачать в

архивной версии, но тогда его нельзя будет изменять через Git (точнее, можно, но будет неудобно, в рамках урока делаем проще, поэтому скопируйте ссылку). Еще ссылку можно скопировать из адресной строки, обычно с ней тоже работает.

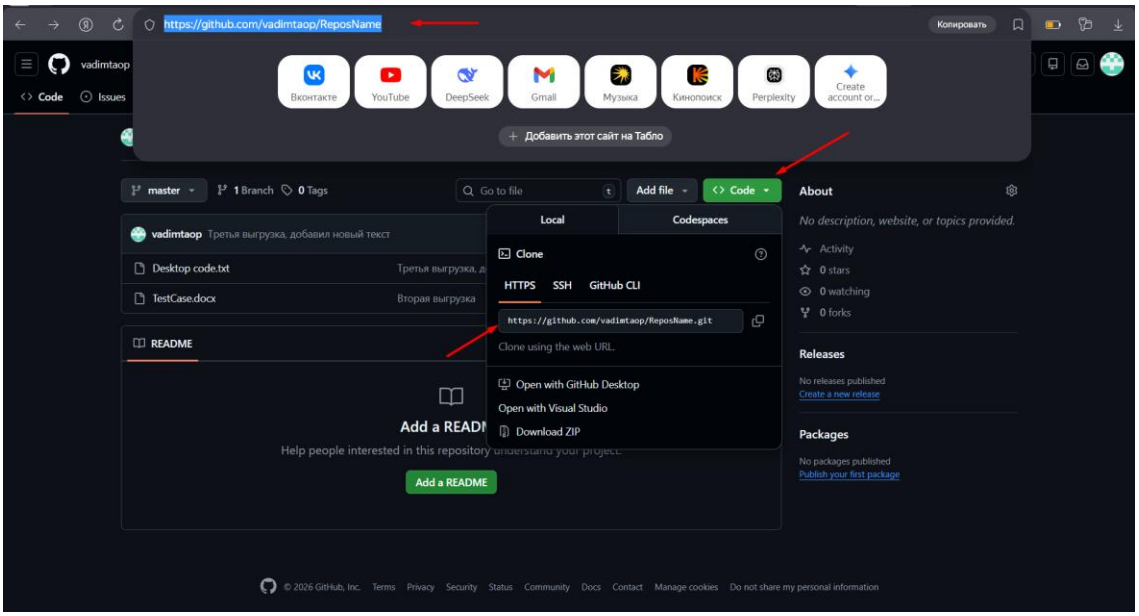

Рисунок 34.

Далее пишем команду в терминале «git clone ссылка» (Рисунок 35.) и ждем сообщение об успешном клонировании.

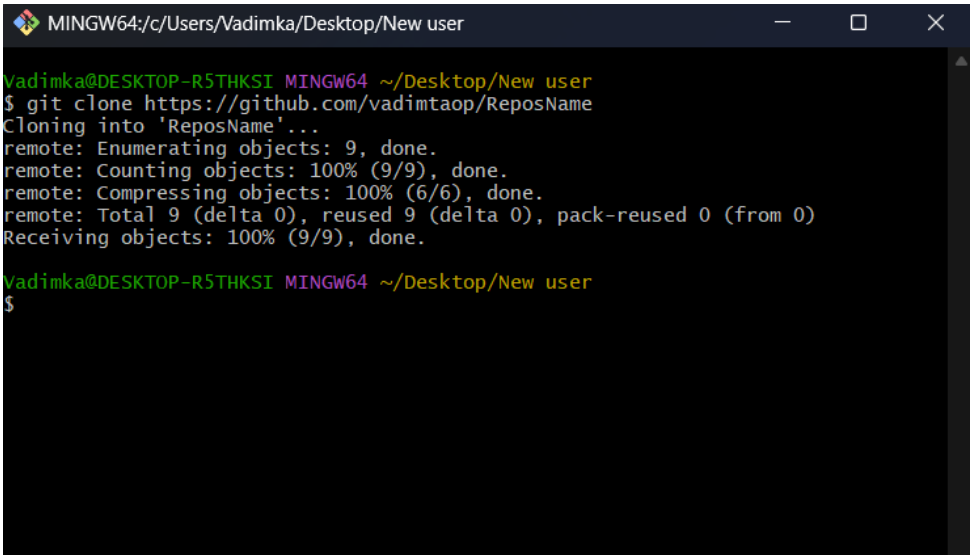

Рисунок 35.

После клонирования убедитесь, что проект правильно скачался (Рисунки 36, 37).

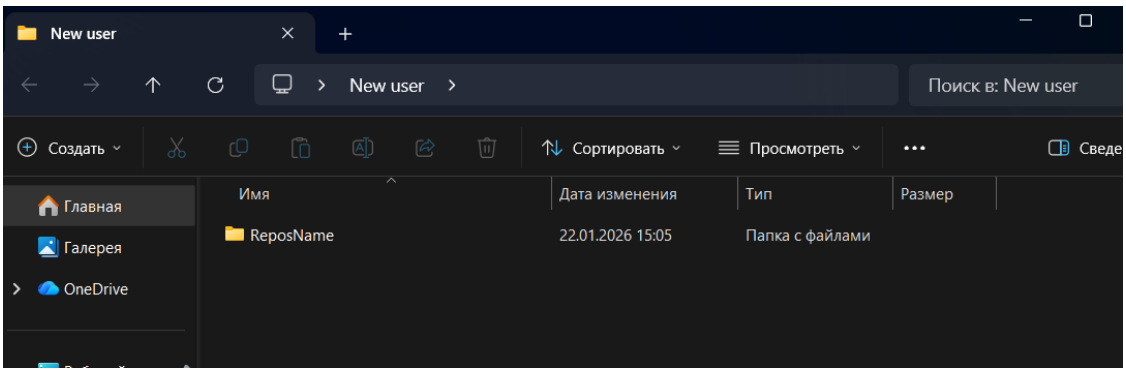

Рисунок 36.

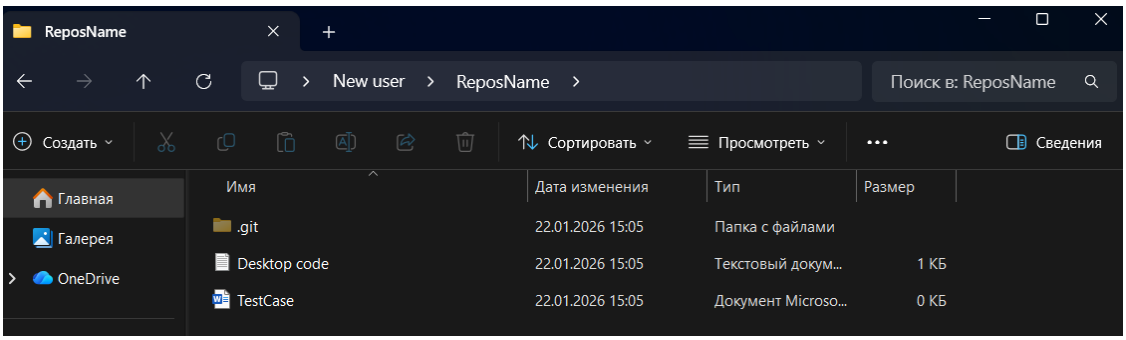

Рисунок 37.

Можем заметить, что проект загрузился в последней версии. Теперь попробуем вернутся в предыдущим версиям проекта. Для начала не забудьте зайти в папку с проектом. Далее нужно узнать хэш сохраненного коммита. Чтобы их узнать используем команду «git log --oneline» (Рисунок 38.).

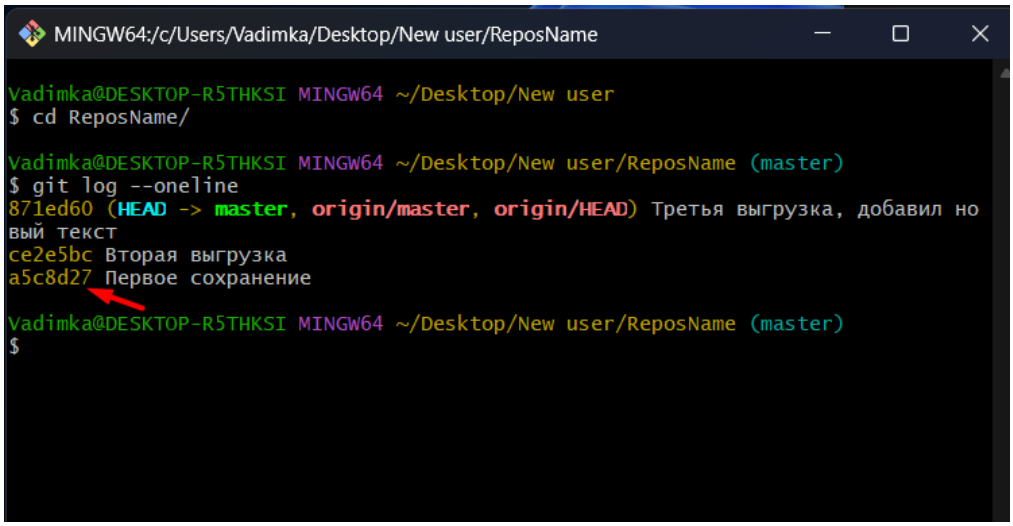

Рисунок 38.

Ответом на эту команду будет список наших коммитов, каждому из которых, присваивается свой хэш. Узнав его, попробуем перейти на предыдущую версию, для этого используем команду «git checkout название»,

указав либо, как в нашем случае хэш, либо название ветки (в данном уроке ветки мы пока, что затрагивать не будем) (Рисунок 39.).

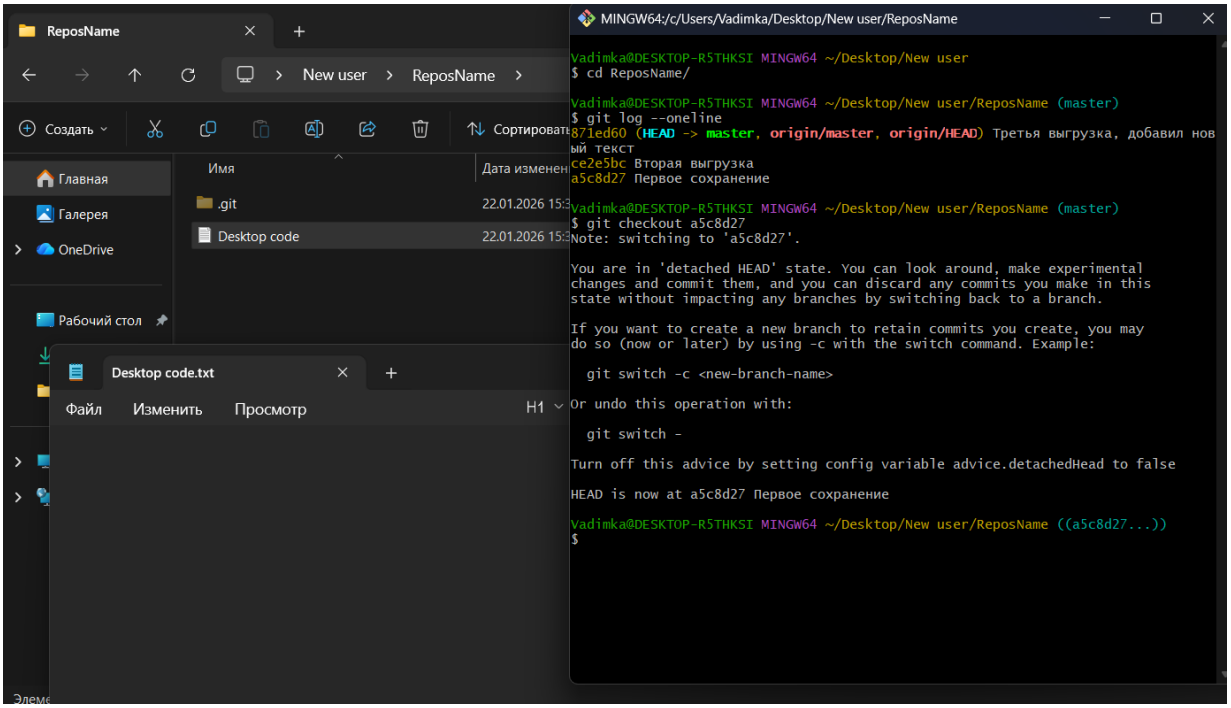

Рисунок 39.

Чтобы вернуться обратно на новую версию, используйте команду «git checkout master».

В этом уроке мы разобрали минимальные команды для использования системы контроля версий. Эти знания будут необходимы для успешной сдачи экзамена. Для серьёзного использования в реальных условиях требуется намного больше знаний и команд, например, понимания ветвей, объединение проектов и так далее.

Если у вас возникло желание изучить Git более подробнее, то существует подробная официальная документация по ссылке https://git-scm.com/book/ru/v2 . А для тех, кто не любит читать существует игра-песочница, где очень удобно визуализируются команды с подробным и простым объяснением по ссылке https://learngitbranching.js.org/?locale=ru_RU .