Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Российская академия народного хозяйства и государственной службы при Президенте Российской Федерации»

Нижегородский институт управления

Кафедра Информатики и информационных технологий

Дисциплина «Программная инженерия»

Отчет о проделанной работе.

Выполнил студент группы Иб-321

Никитин Михаил Алексеевич

Нижний Новгород

2026 г.

Оглавление

[Цель работы 3](#_Toc221996845)

[Оборудование и ПО 3](#_Toc221996846)

[Теоретическая часть 3](#_Toc221996847)

[Практическая часть 7](#_Toc221996848)

[Контрольные вопросы 14](#_Toc221996849)

[Вывод 14](#_Toc221996850)

**Изучение работы с удалёнными и локальными репозиториями**

Цель работы: Освоить базовые операции работы с системой контроля версий Git и облачным сервисом GitHub: регистрацию, создание репозитория, настройку SSH‑ключей, базовые команды Git, редактирование файлов и управление изменениями.

# Оборудование и ПО

* Персональный компьютер с ОС Windows 11 Pro
* Установленный Git
* Браузер Yandex
* Терминал/командная строка

# Теоретическая часть

**Типы Git**

1. **Локальные системы контроля версий (Local VCS):** Это самый простой подход, при котором все изменения хранятся в локальной базе данных на вашем компьютере. Примером может служить программа RCS. Однако, этот подход не подходит для командной работы, так как хранит историю изменений только на одном устройстве .
2. **Централизованные системы контроля версий (Centralized VCS — CVCS):** В таких системах (например, Subversion или CVS) есть один главный сервер, который хранит все версии файлов. Разработчики подключаются к нему, чтобы получить нужную версию или сохранить свои изменения. Это упрощает совместную работу, но имеет критический недостаток: если сервер выйдет из строя или повредится диск, история проекта будет потеряна .
3. **Распределенные системы контроля версий (Distributed VCS — DVCS):** Главное отличие в том, что клиенты (разработчики) копируют себе не только последнюю версию файлов, но **полностью весь репозиторий**, включая всю историю изменений. Это делает систему очень надежной: при сбое на сервере любой клиентский репозиторий может быть скопирован обратно для восстановления данных .

**Назначение Git**

**Git** — это распределенная система контроля версий, изначально созданная Линусом Торвальдсом в 2005 году для разработки ядра Linux . Ее главное назначение — отслеживание изменений в файлах и координация работы над проектами нескольких человек .

**Задачи, которые решает Git:**

* **Контроль версий:** Позволяет фиксировать изменения, чтобы в любой момент можно было вернуться к предыдущему состоянию проекта или конкретного файла .
* **Совместная работа:** Дает возможность нескольким разработчикам одновременно работать над одним проектом, а затем объединять их изменения в единую кодовую базу .
* **Параллельная разработка:** С помощью механизма **веток (branches)** разработчики могут создавать изолированные линии разработки для новых функций или экспериментов, не мешая работе над основной версией проекта .
* **Надежность и целостность:** Каждое изменение в Git снабжается контрольной суммой, что гарантирует целостность истории. Кроме того, благодаря распределенной архитектуре, потерять данные проекта очень сложно .

**Основные понятия (термины)**

* **Репозиторий (Repository):** Хранилище вашего проекта, которое содержит все файлы, папки и полную историю их изменений.
* **Коммит (Commit):** Фиксация изменений в репозитории. Это как "снимок" состояния проекта на определенный момент времени с комментарием о том, что было сделано .
* **Ветка (Branch):** Независимая линия разработки. По умолчанию основная ветка называется main (или master).
* **Индекс / Область подготовленных файлов (Index / Staging Area):** Промежуточная зона, где вы собираете изменения перед тем, как зафиксировать их в коммите .
* **Удаленный репозиторий (Remote Repository):** Копия вашего репозитория, хранящаяся на сервере (например, на GitHub). Он используется для синхронизации работы между участниками команды .

**Основные команды Git**

Команды Git можно разделить на несколько логических групп в зависимости от того, для чего они предназначены .

*1. Настройка и создание репозитория*

| **Команда** | **Назначение** | **Пример использования** |
| --- | --- | --- |
| git config | Настройка параметров Git (имя пользователя, email) . | git config --global user.name "Ваше Имя" |
| git init | Создание (инициализация) нового локального репозитория в текущей папке . | git init |
| git clone | Копирование существующего удаленного репозитория на свой компьютер . | git clone <url-репозитория> |

*2. Работа с изменениями*

| **Команда** | **Назначение** | **Пример использования** |
| --- | --- | --- |
| git status | Показывает текущее состояние файлов: какие изменены, какие готовы к коммиту . | git status |
| git add | Добавляет файлы в индекс (staging area), подготавливая их к коммиту . | git add README.md или git add .(добавить всё) |
| git commit | Фиксирует изменения, находящиеся в индексе, создавая новый коммит . | git commit -m "Сообщение с описанием изменений" |
| git diff | Показывает разницу между версиями файлов (рабочей и в индексе, или между коммитами) . | git diff |
| git rm | Удаляет файл из рабочей копии и индекса . | git rm old\_file.txt |
| git mv | Перемещает или переименовывает файл . | git mv old\_name.txt new\_name.txt |
| git clean | Удаляет неотслеживаемые файлы из рабочей директории . | git clean -f |

*3. Работа с ветками*

| **Команда** | **Назначение** | **Пример использования** |
| --- | --- | --- |
| git branch | Просмотр списка веток или создание новой . | git branch (список) или git branch new-feature (создать) |
| git checkout / git switch | Переключение между ветками или восстановление файлов . | git checkout main или git switch new-feature |
| git merge | Слияние изменений из одной ветки в текущую . | git merge new-feature (находясь в ветке main) |

*4. Работа с удаленными репозиториями*

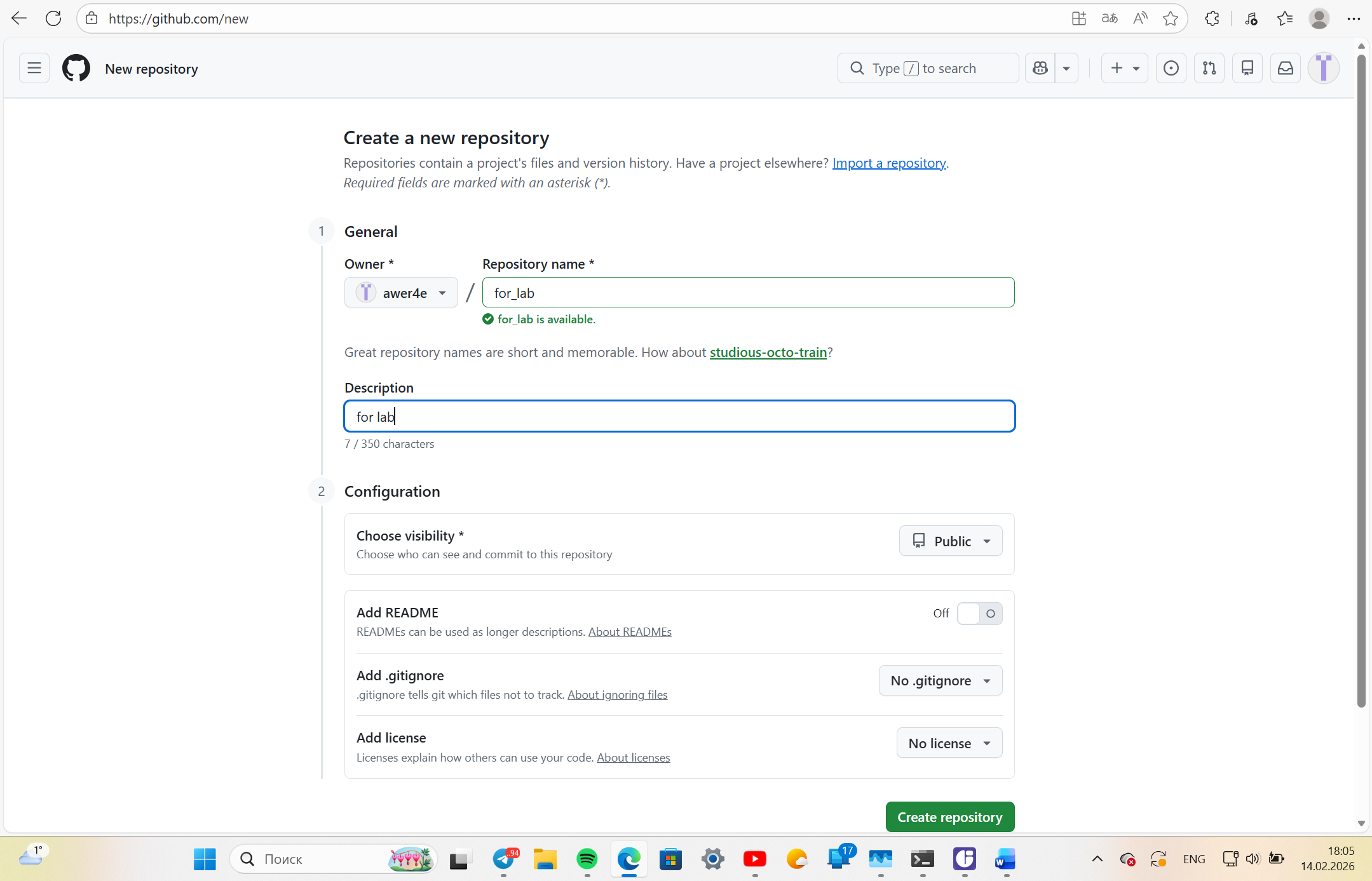
| **Команда** | **Назначение** | **Пример использования** |
| --- | --- | --- |
| git remote | Управление списком удаленных репозиториев . | git remote add origin <url> (добавить связь) |
| git push | Отправка закоммиченных изменений из локального репозитория в удаленный . | git push origin main |
| git pull | Загрузка изменений из удаленного репозитория и слияние их с вашей текущей локальной веткой . | git pull origin main |
| git fetch | Загрузка изменений из удаленного репозитория, но без автоматического слияния . | git fetch origin |

*5. Просмотр истории и отмена изменений*

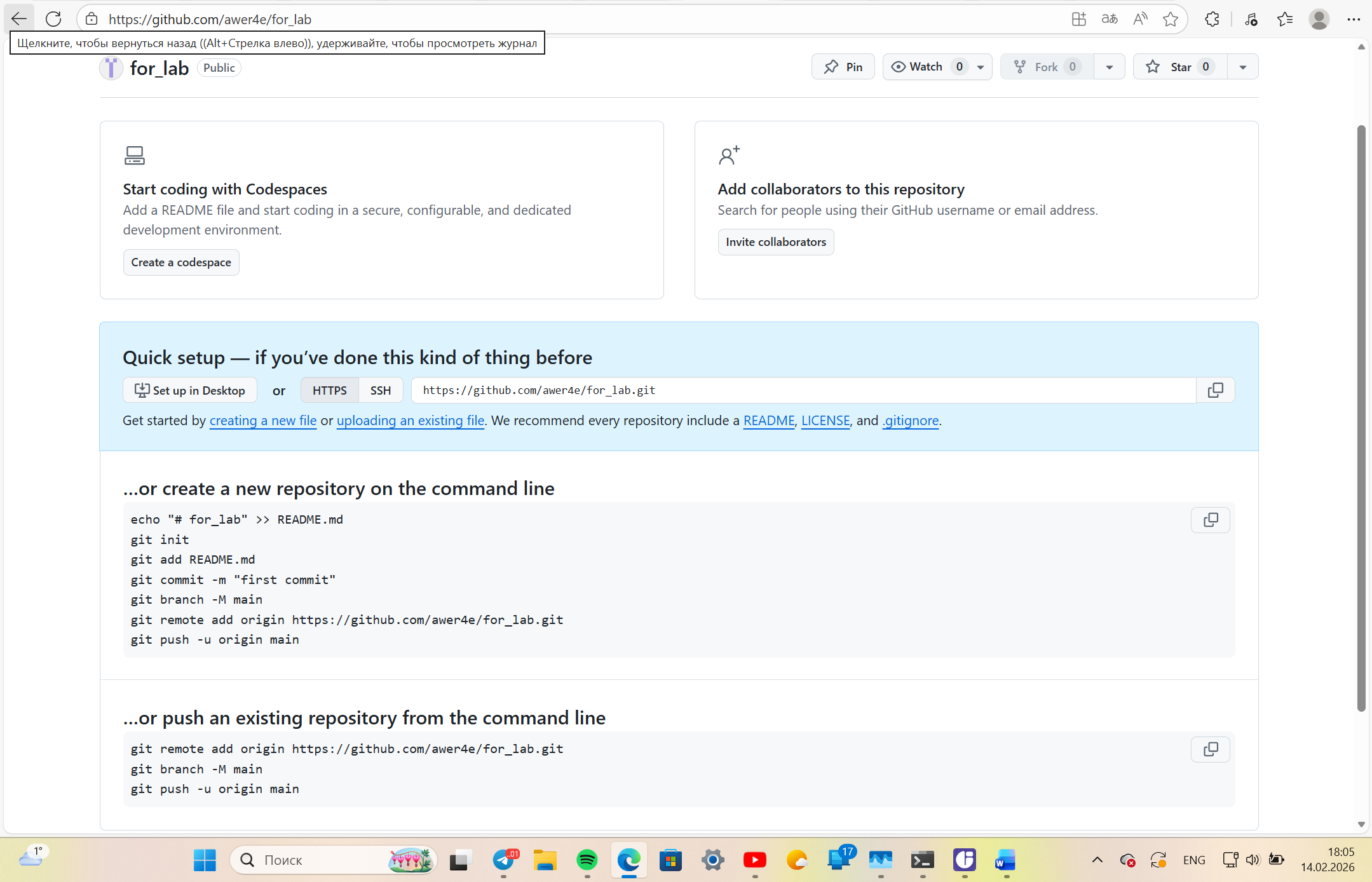
| **Команда** | **Назначение** | **Пример использования** |
| --- | --- | --- |
| git log | Показывает историю коммитов в репозитории . | git log --oneline (краткий формат) |
| git show | Отображает информацию о конкретном коммите . | git show <идентификатор\_коммита> |
| git reset | Отмена изменений. Может использоваться для отмены индексации файла, отката коммитов и т.д. . | git reset --hard HEAD~1 (откатиться на один коммит назад) |
| git restore | Более новая и безопасная команда для отмены изменений в рабочей директории или индексе . | git restore <file> |

# Практическая часть

Я вошел на GitHub, и создал репозиторий.

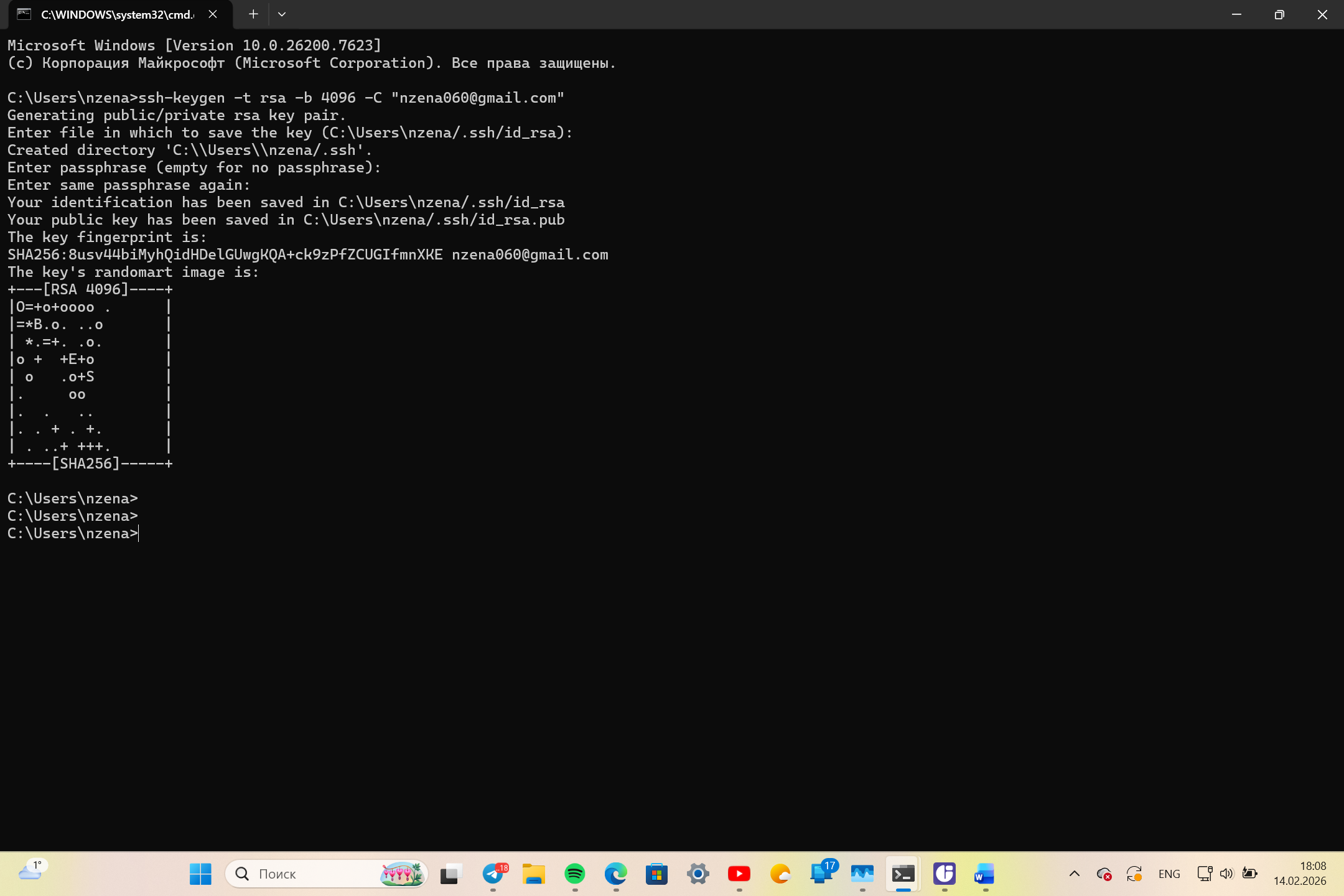


Репозиторий создан.

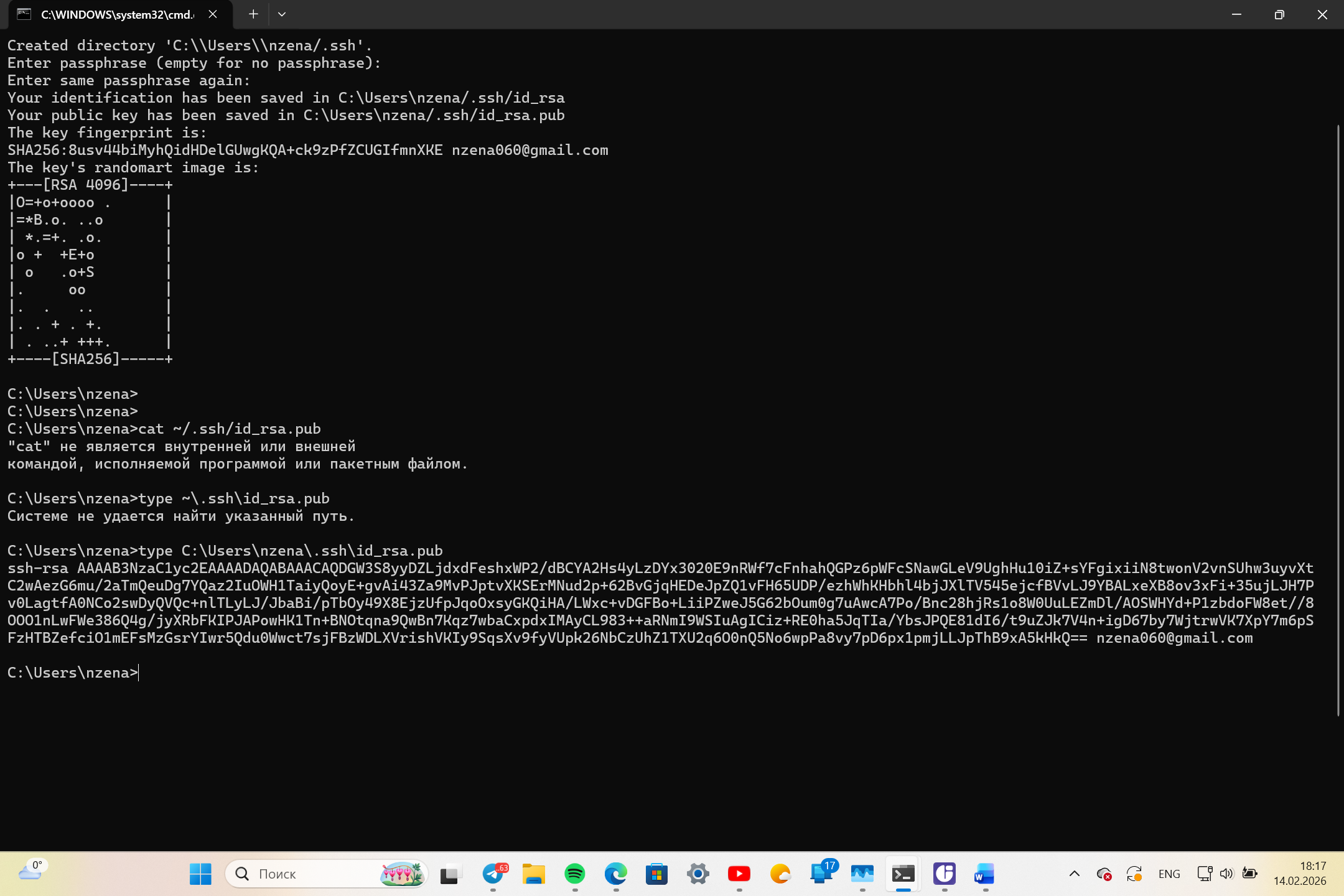


Настройка SSH‑ключа (для безопасной аутентификации)

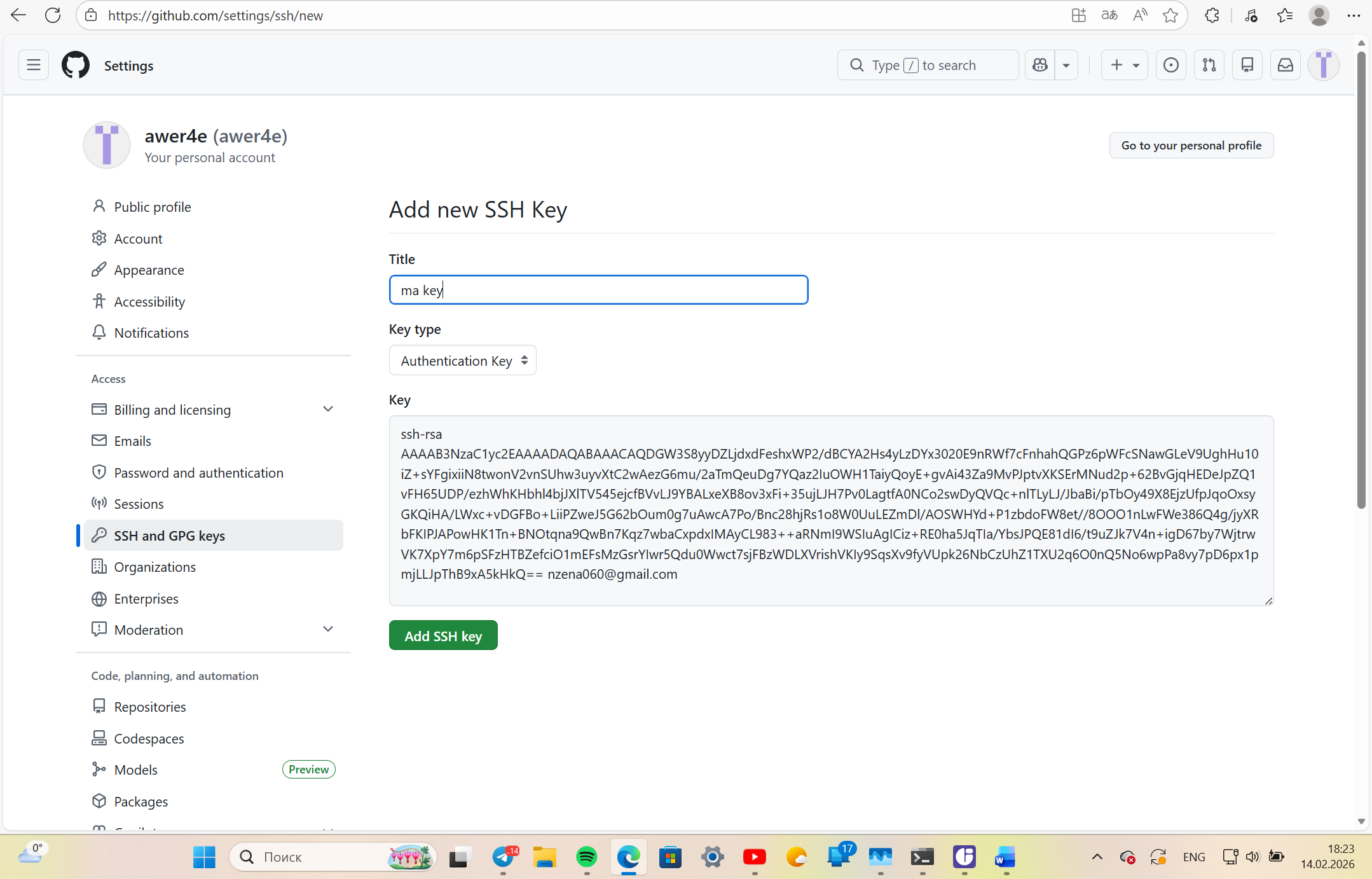
В командной строке вставил команду для создания SSH ключа.



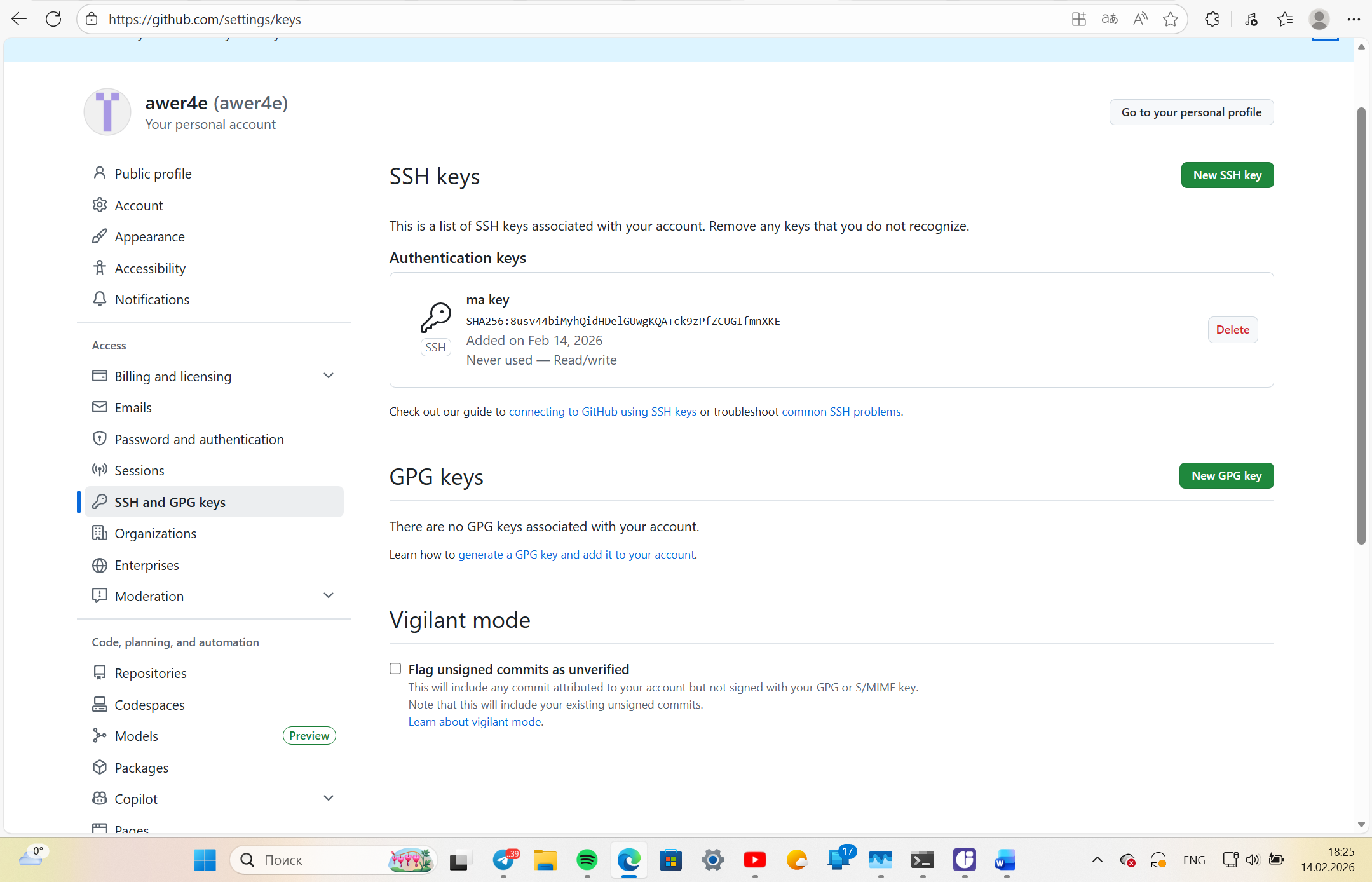
Вывел ключ и скопировал его.



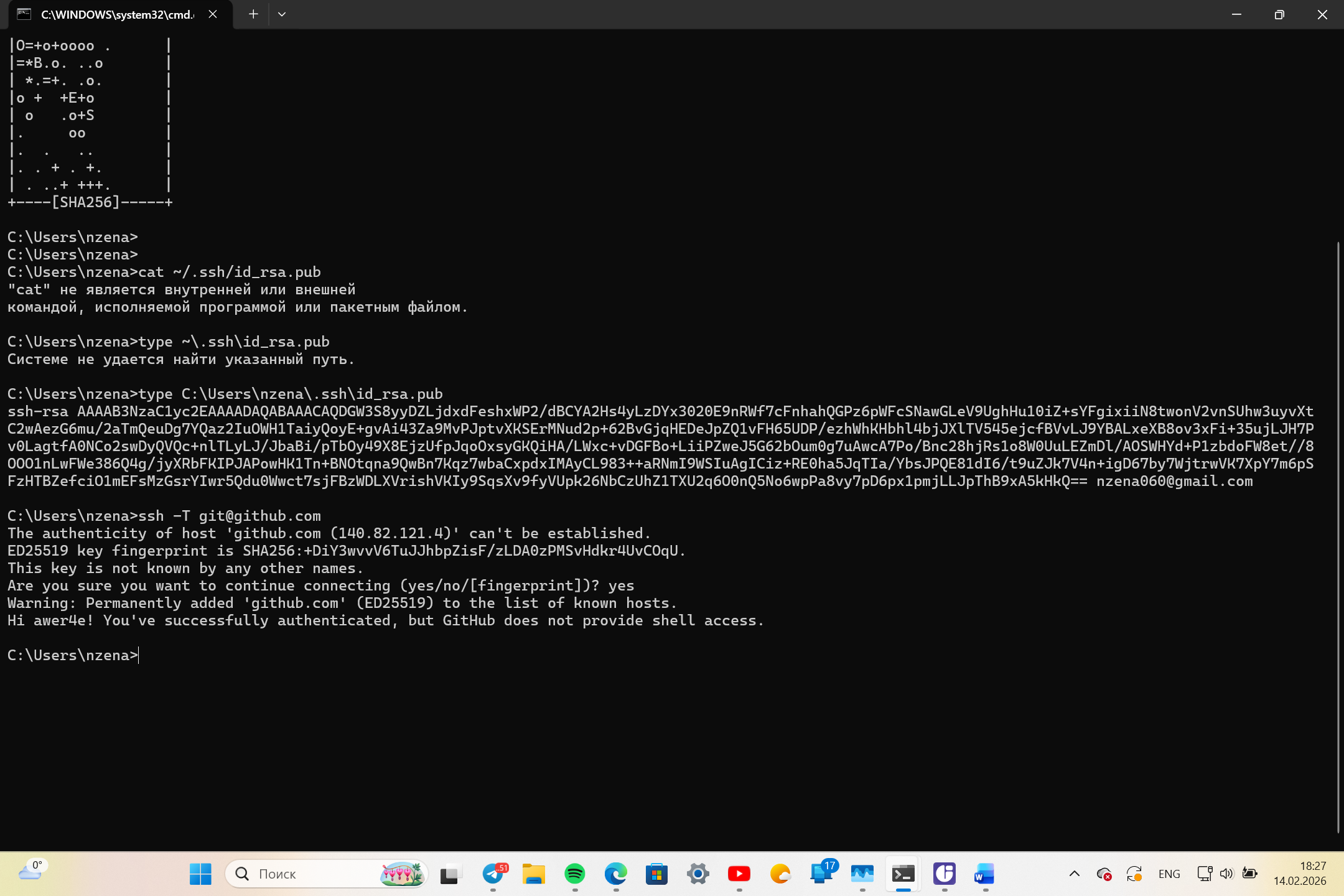
Вставил его в поле на сайте.



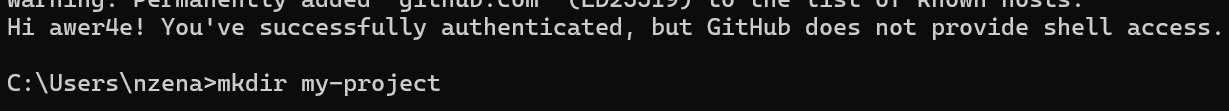
Сохранил изменения.

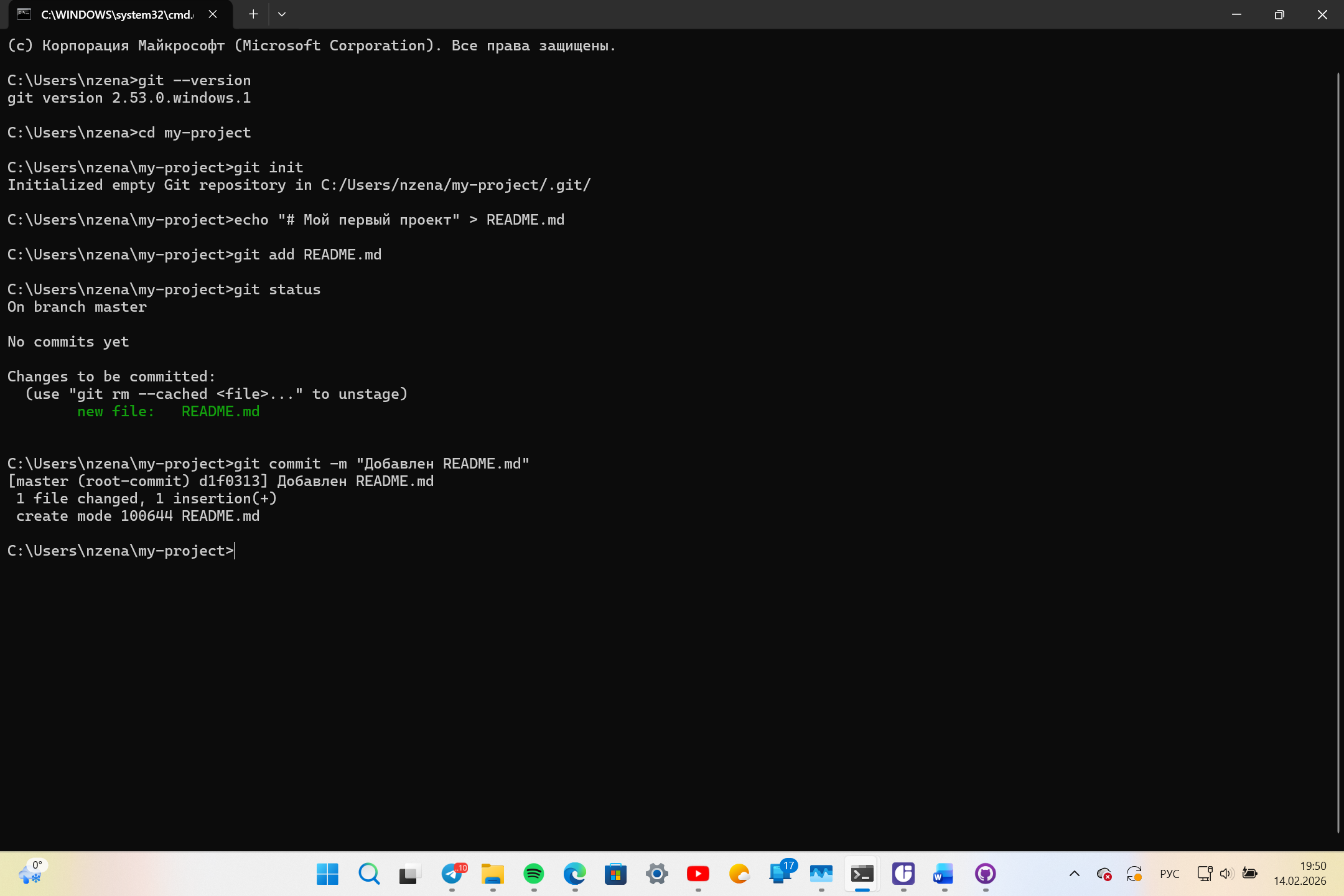


Ключ успешно подключен.

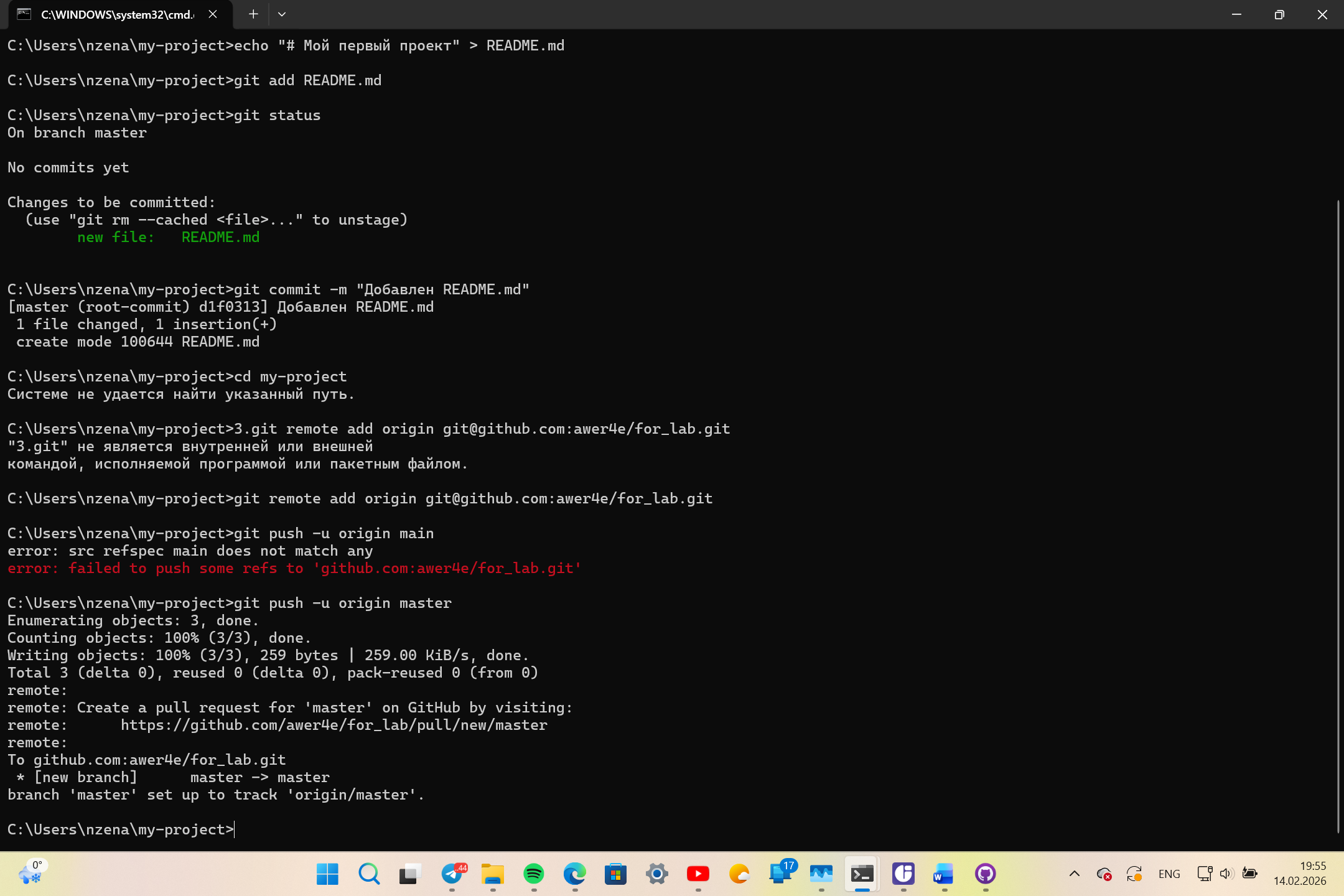


Создал новую папку

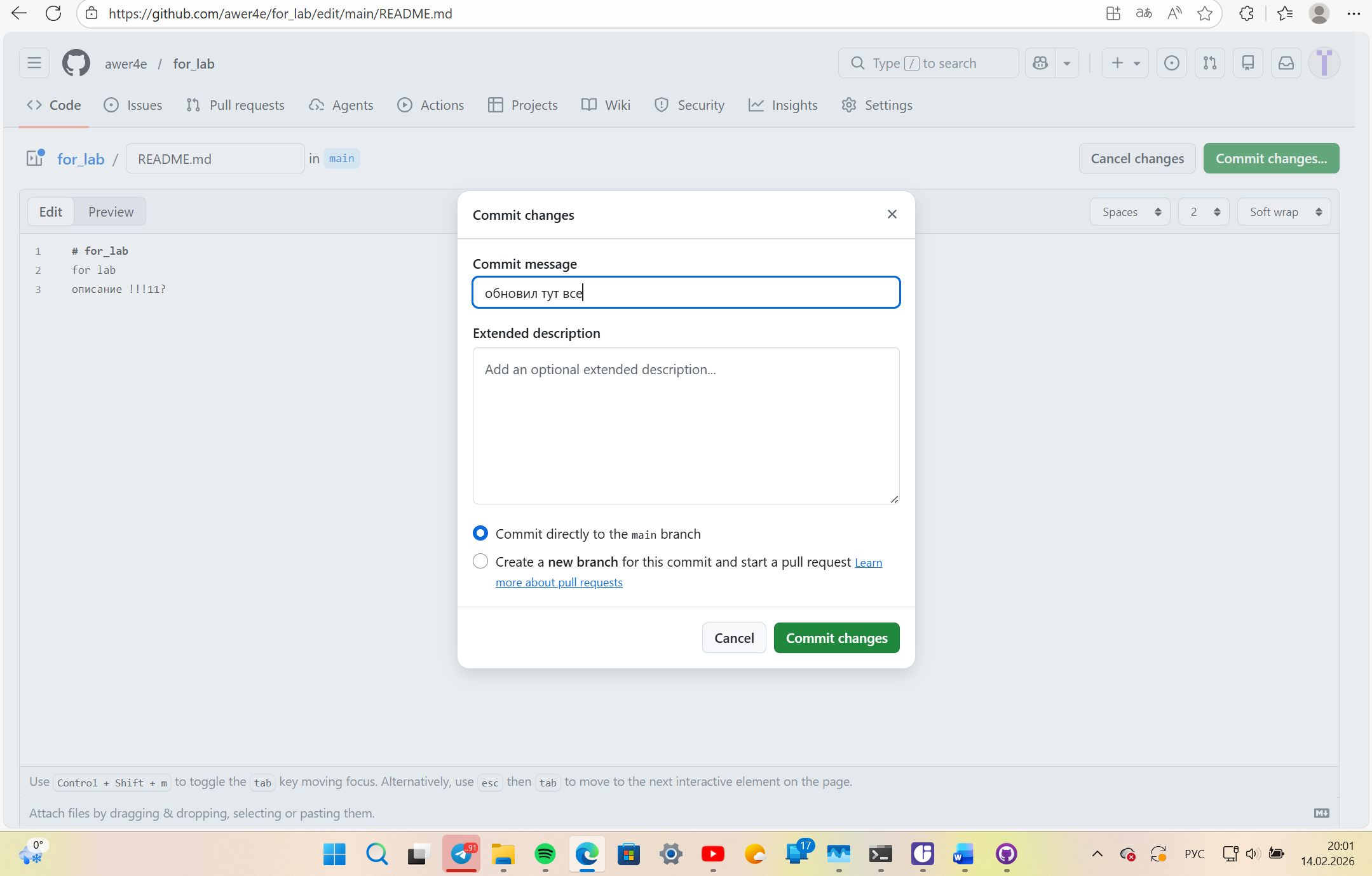




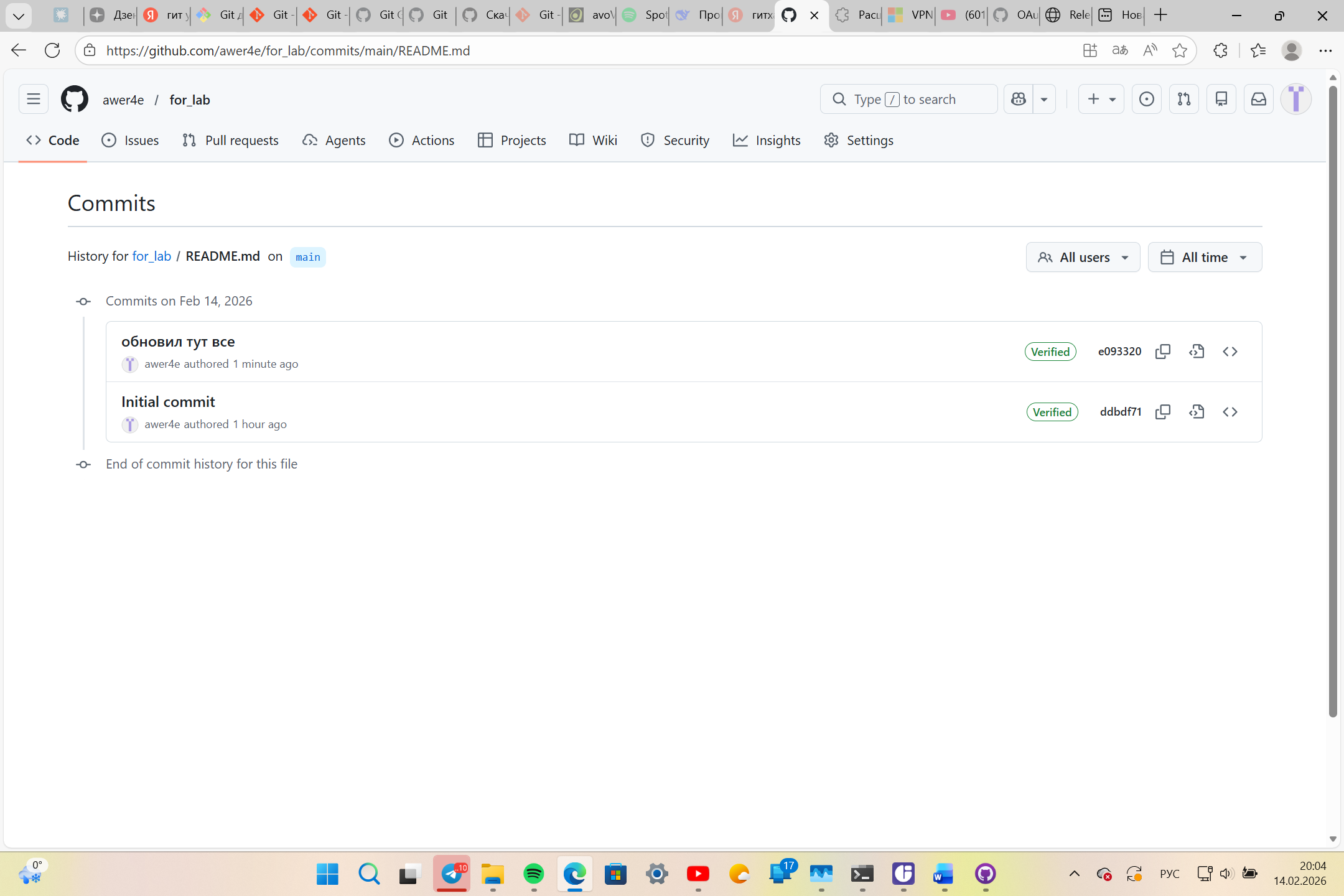
Перешел в эту папку и добавил файл README.



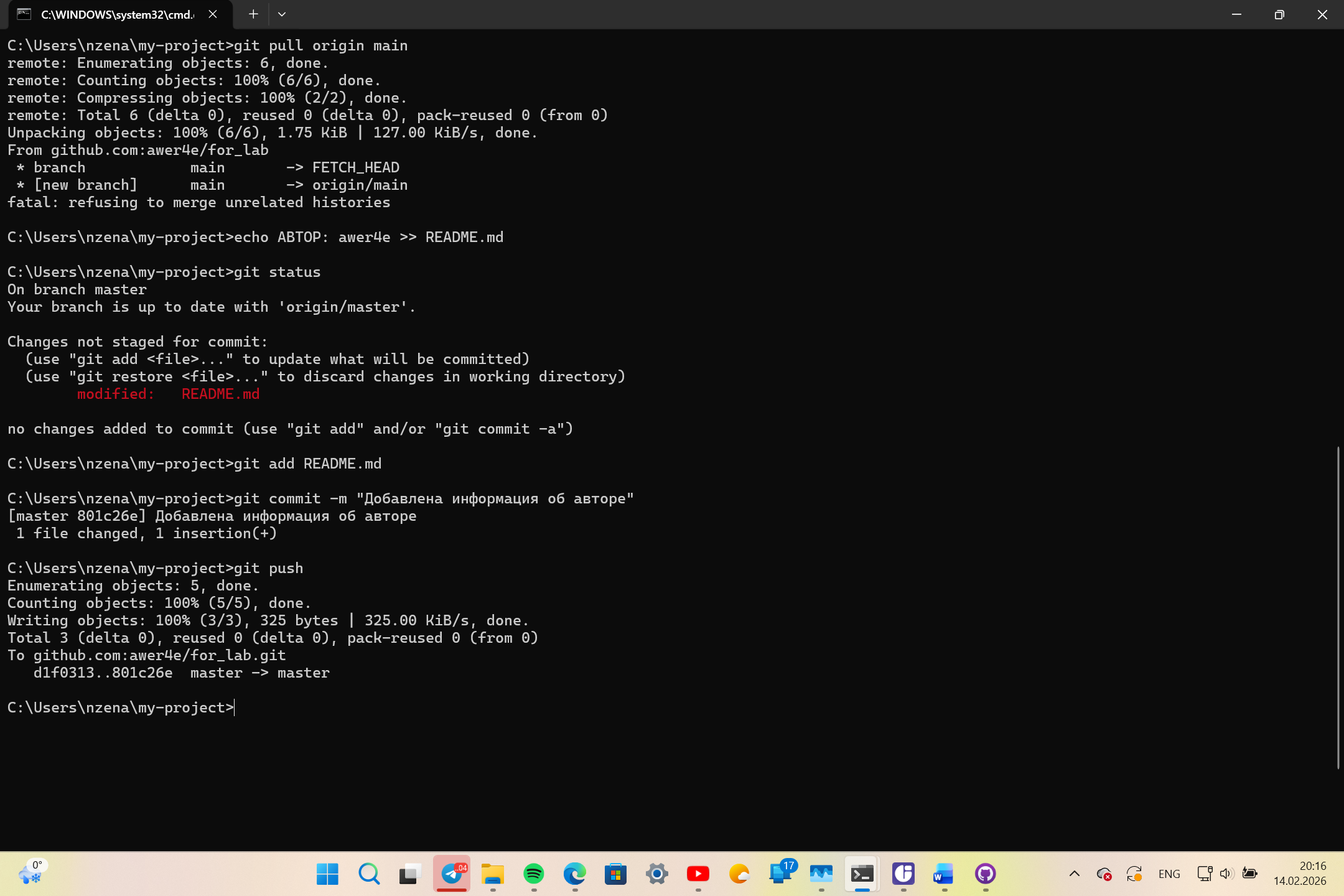
Подключил репозиторий и отправил изменения.



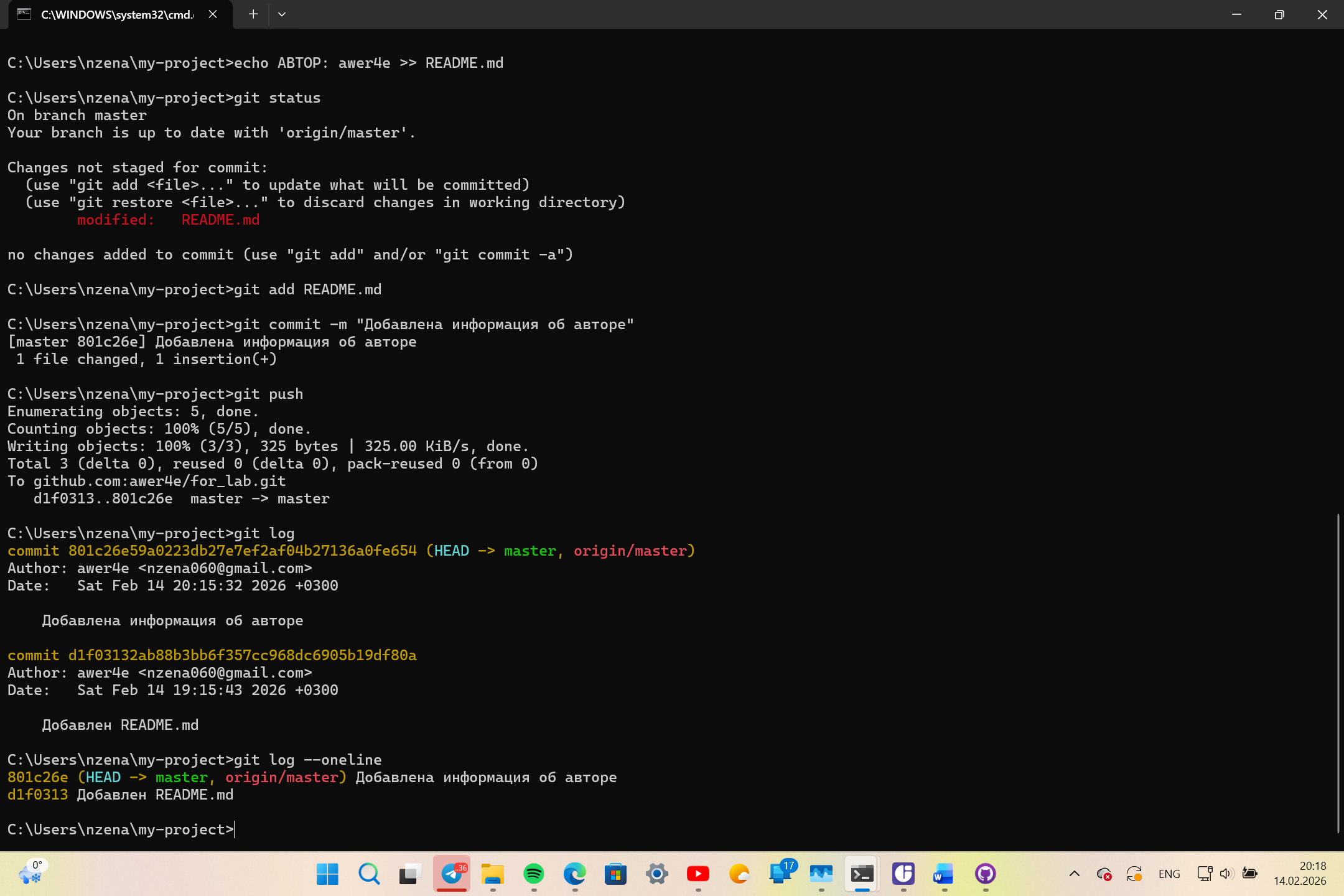
Обновил файл, добавил сообщение, сохранил изменения.



Все сохранилось.



Загрузил изменения git, изменил файл.



Посмотрел изменения.

# Контрольные вопросы

1. Что такое репозиторий?

Репозиторий — это специализированное хранилище проекта, которое включает в себя не только текущий набор файлов и директорий, но и полную историю их изменений (метаданные). Репозиторий позволяет отслеживать версии кода, возвращаться к предыдущим состояниям проекта и координировать работу нескольких разработчиков над общими файлами.

2. Для чего нужен SSH‑ключ при работе с GitHub?

SSH-ключ используется для обеспечения безопасной аутентификации между локальной машиной разработчика и сервером GitHub без необходимости постоянного ввода логина и пароля. Использование пары ключей (открытого и закрытого) позволяет серверу однозначно идентифицировать пользователя по зашифрованному каналу, что повышает уровень безопасности и автоматизирует процесс обмена данными.

3. В чём разница между git commit и git push?

Разница заключается в области применения команды:

git commit: Фиксирует проиндексированные изменения в локальном репозитории. Создается новый снимок (snapshot) состояния проекта, доступный только на текущей рабочей станции.

git push: Переносит зафиксированные локальные коммиты в удаленный (серверный) репозиторий. Только после выполнения этой команды изменения становятся доступными для других участников проекта.

4. Как просмотреть список изменённых файлов перед коммитом?

Для просмотра текущего состояния рабочей директории и индекса используется команда:

git status

Она выводит список файлов, которые были изменены, удалены или еще не отслеживаются системой Git. Для просмотра конкретных изменений внутри самих файлов (разницы строк) используется команда:

git diff

5. Что означает команда git pull?

Команда git pull используется для получения последних изменений из удаленного репозитория и их немедленного объединения с текущей локальной веткой. Технически она представляет собой последовательное выполнение двух операций:

git fetch (загрузка данных с сервера в локальную копию).

git merge (слияние полученных правок с текущим рабочим кодом).

Вывод:

В ходе выполнения лабораторной работы я освоил основные принципы работы с системой контроля версий Git и облачным сервисом GitHub. Я научился создавать и настраивать репозитории, генерировать и использовать SSH-ключи для безопасного подключения, выполнять базовые команды, а также синхронизировать локальный и удалённый репозитории. Кроме того, я получил навыки редактирования файлов непосредственно на GitHub и работы с историей изменений.