**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА**

**И ГОСУДАРСТВЕННОЙ СЛУЖБЫ**

**ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**НИЖЕГОРОДСКИЙ ИНСТИТУТ УПРАВЛЕНИЯ – филиал РАНХиГС**

Факультет: Управления

Кафедра: информатика и информационные технологии

Направление подготовки / специальность: Прикладная информатика (09.03.03)

*(код, наименование)*

Направленность (профиль) / специализация: Корпоративные информационные системы

управления

**ПРАКТИЧЕСКАЯ РАБОТА**

по дисциплине: Информатика и Программирование

на тему: Git

**АВТОР**

Обучающийся курса группы ИБ-321

очной формы обучения

Беляев Е. С.

**РУКОВОДИТЕЛЬ**

Окулич Евгения Викторовна

*(ученая степень, ученое звание)*

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г.

*(дата защиты)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись) (фамилия, инициалы)*

Нижний Новгород,2025

**Оглавление**

[Цель работы 2](#_Toc1605064979)

[Отчет 3](#_Toc835125526)

[Назначение, возможности и особенности 4](#_Toc637248396)

[Сравнение Git, GitHub, GitLab 5](#_Toc551479236)

[Российская система GitVerse 5](#_Toc1314415033)

[Команды системы Git 7](#_Toc1005550483)

[Вывод 15](#_Toc210258164)

# **Цель работы**

Познакомиться с такими системами как Git, GitHub, GitLab, GitVerse, а также научиться работать с Git репозиториями и изучить основные команды для работы с Git.

# **Отчет**

## **Назначение, возможности и особенности**

**Git**:

* **Назначение**: Это распределённая система контроля версий, которая позволяет отслеживать изменения в коде и управлять версиями проектов. Git дает возможность работать над проектами одновременно нескольким разработчикам.
* **Возможности**: Git позволяет создавать ветки (branches), что позволяет разработчикам работать над новыми функциями или исправлениями без риска повредить основную кодовую базу. Также предоставляет инструменты для слияния (merge) изменений, отката (revert) коммитов, а также просмотр истории изменений.
* **Особенности**: Git сохраняет полную историю изменений в локальном репозитории, что позволяет разработчикам работать оффлайн. Он быстро и эффективно управляет большими проектами.

**GitHub**:

* **Назначение**: Это веб-сервис для хостинга проектов, использующих Git. GitHub предоставляет платформу для совместной разработки и обмена кодом.
* **Возможности**: Пользователи могут создавать репозитории, вносить изменения через pull requests, управлять задачами с помощью issues, обсуждать коды, а также использовать дополнительные инструменты, такие как GitHub Actions для автоматизации.
* **Особенности**: GitHub известен своей социальной составляющей: разработчики могут следить за работой других, комментировать изменения и предлагать улучшения. Платформа имеет бесплатные и платные тарифы, поддерживающие открытое и закрытое программное обеспечение.

**GitLab**:

* **Назначение**: Это тоже веб-сервис для управления репозиториями Git, но с более широкими возможностями, включая CI/CD (непрерывная интеграция и доставка).
* **Возможности**: GitLab предлагает функционал для управления проектами, автоматизации тестирования и развертывания, а также интеграцию с DevOps-практиками. Поддерживает создание точных метрик и анализа командной работы.
* **Особенности**: GitLab предоставляет более мощные инструменты для управления проектами и командой по сравнению с GitHub, включая встроенные инструменты для разработки и развертывания программного обеспечения.

## **Сравнение Git, GitHub, GitLab**

|  |  |  |  |
| --- | --- | --- | --- |
| **Параметр** | **Git** | **GitHub** | **GitLab** |
| **Тип** | Система контроля версий | Хостинг для репозиториев Git | Хостинг для репозиториев Git с дополнительными функциями |
| **Назначение** | Управление версиями кода | Совместная разработка и обмен кодом | Управление проектами и полные CI/CD |
| **Интерфейс** | Командная строка | Веб-интерфейс | Веб-интерфейс |
| **Социальные функции** | Нет | Да, комментирование, подписка | Да, включены инструменты для управления проектами |
| **CI/CD** | Нет | Ограниченные возможности | Да, встроенные инструменты для CI/CD |
| **Лицензия** | Открытый | Бесплатный и платный | Открытый и платный |
| **Подход к репозиториям** | Распределённый | Централизованный | Централизованный и поддержка распределённых репозиториев |

## **Российская система GitVerse**

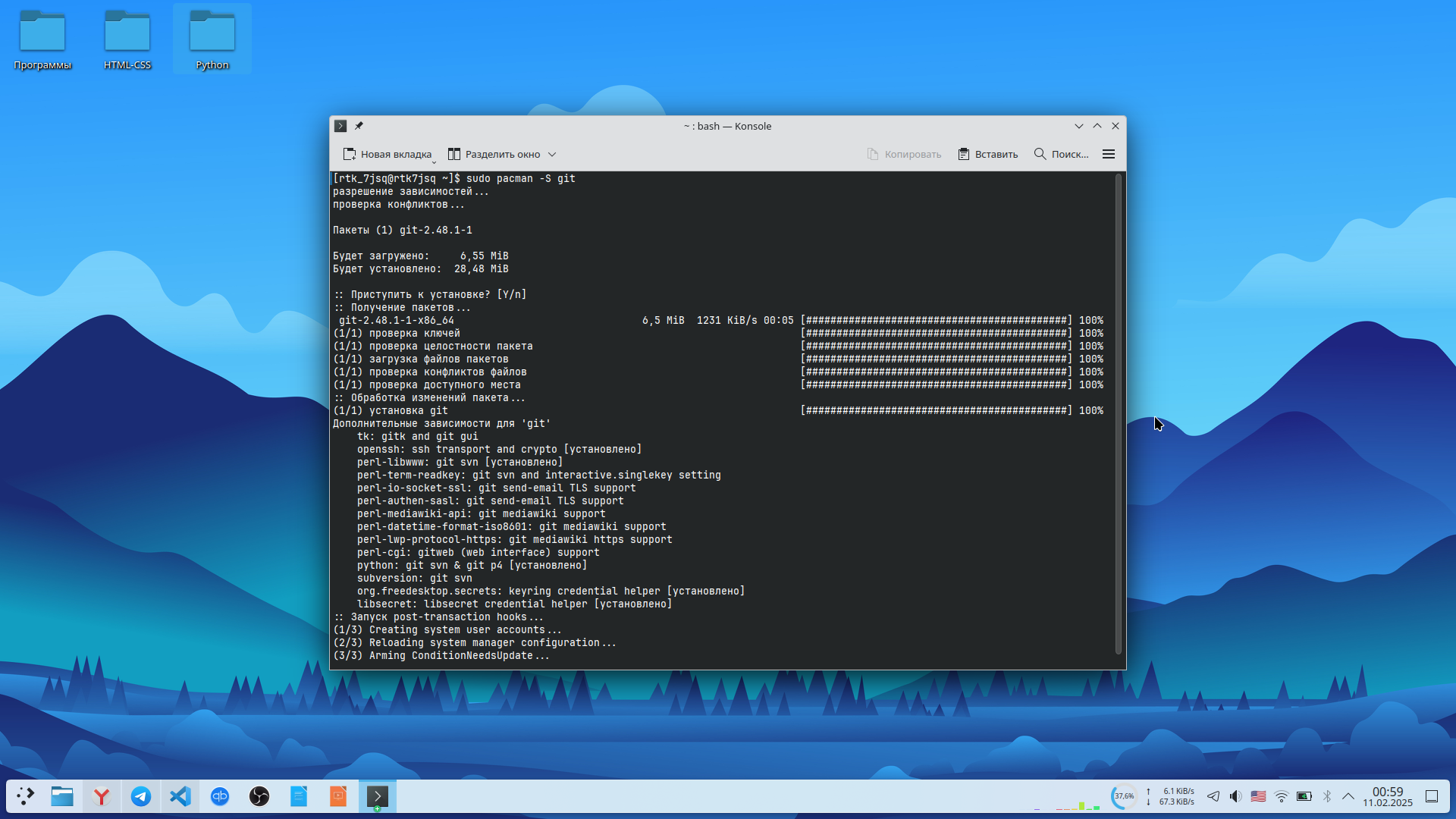
**GitVerse** — это российская платформа для хостинга Git-репозиториев и управления проектами, созданная с целями, аналогичными GitHub и GitLab. Отличия от GitHub минимальные:

1. **Локализация**:
   * **GitVerse**: Ориентирован на российский рынок, имеет русскоязычный интерфейс и поддержку, что может быть удобнее для пользователей из России.
   * **GitHub**: Интерфейс на английском языке, хотя есть поддержка других языков, общеевропейская и международная аудитория в целом.
2. **Соблюдение законодательства**:
   * **GitVerse**: Разработан с учетом российского законодательства и стандартов, что может быть важным для организаций, работающих в России.
   * **GitHub**: Ведется согласно международным стандартам и законодательству стран, в которых он разрабатывается и используется.
3. **Интеграция с местными сервисами**:
   * **GitVerse**: Внедрение и интеграция с популярными российскими разработками и системами может быть более предпочтительным.
   * **GitHub**: Интеграция с международными сервисами и системами, используемыми по всему миру.
4. **Функциональность**:
   * **GitVerse**: Предлагает базовые функции для работы с Git, но может иметь менее развитые инструменты для CI/CD и автоматизации по сравнению с GitHub.
   * **GitHub**: Судя по своему статусу, предлагает широкий спектр функций, включая GitHub Actions для автоматизации, поддержку пакетов и сложные возможности интеграции.
5. **Сообщество и поддержка**:
   * **GitVerse**: Фокус на российском сообществе разработчиков, что может способствовать обмену опытом и лучшими практиками в рамках страны.
   * **GitHub**: Имеет большое международное сообщество, и поддержка часто осуществляется на английском языке.

## **Команды системы Git**

Чтобы начать работу с Git, нужно его установить, для этого в консоли пишем следующую команду и ожидаем установку пакета:

**sudo pacman –S git**

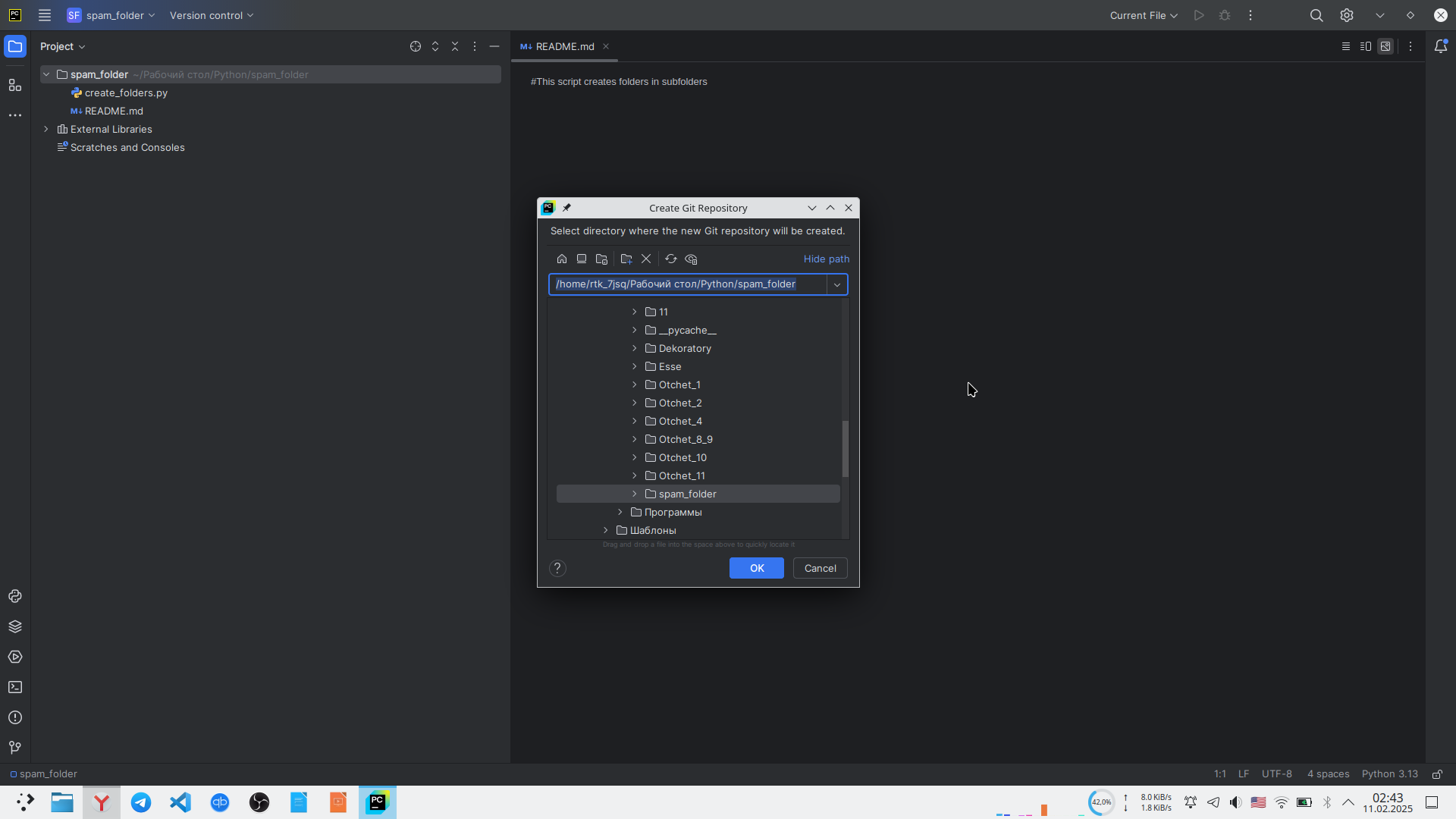
Рисунок 1

Теперь настроим имя пользователя и почту, которые будет указываться при создании коммитов, для этого введем следующие команды:

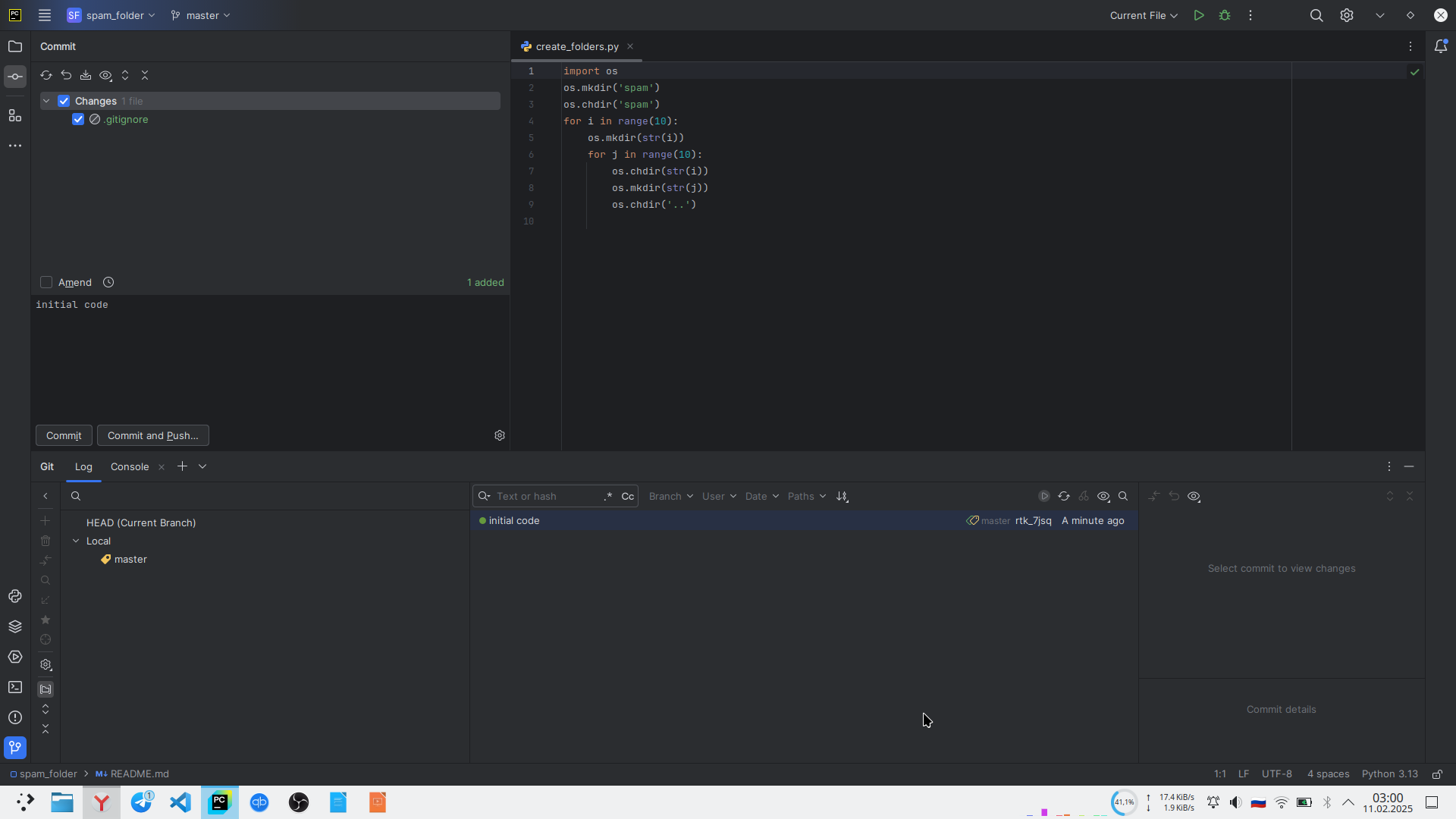
**git config --global user.name "rtk\_7jsq"**

**git config –global user.email "address@email.com"**

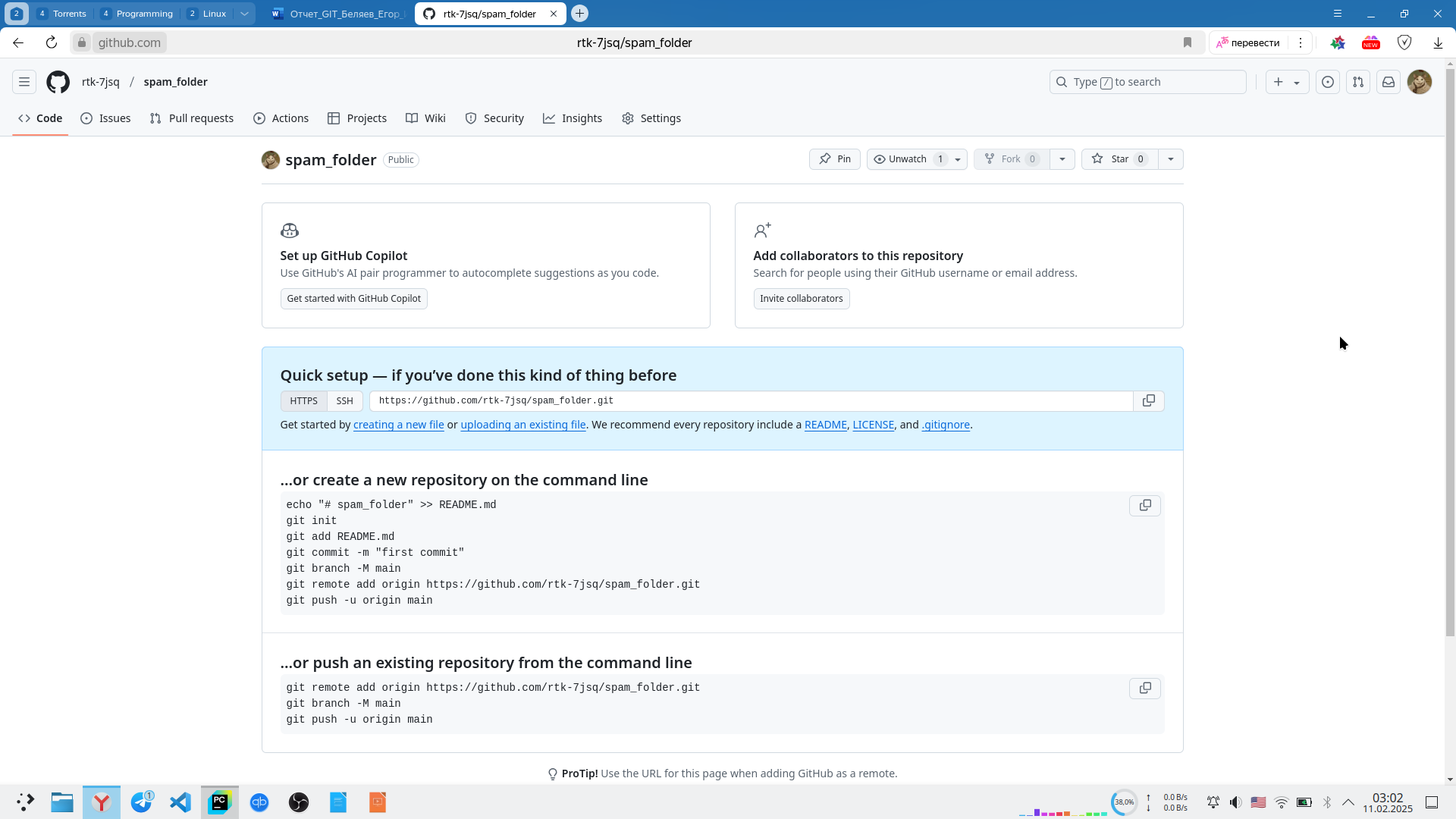
Теперь создадим наш локальный Git репозиторий для этого заходим в папку проекта в Pycharm, сверху нажимаем Version Control > Create Git Repository, появляется меню, в котором выбираем папку с нашим проектом:

Рисунок 2

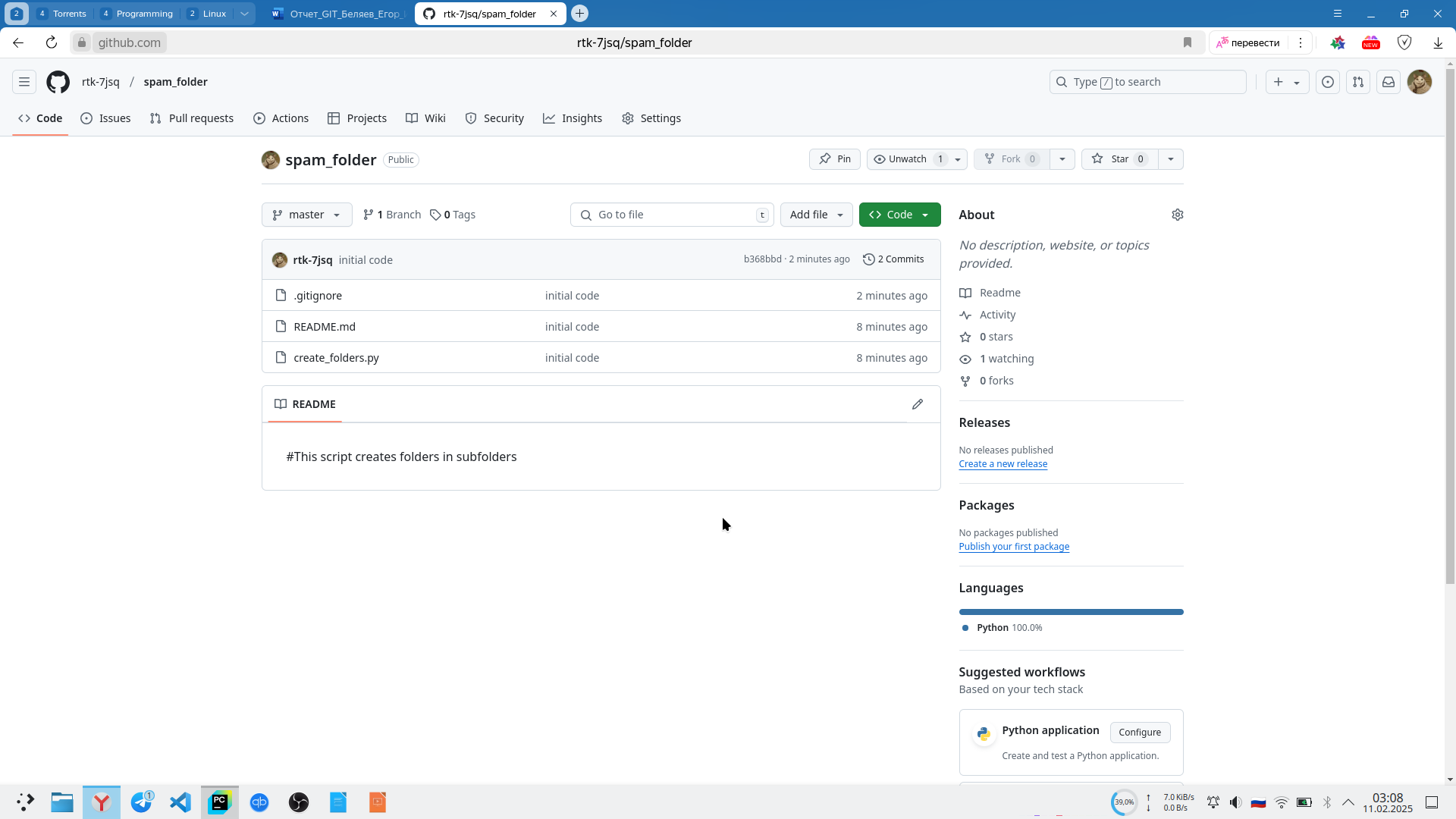
Теперь создадим первый коммит, для этого выбираем слева меню Commit и выбираем все файлы нашего проекта, после чего нажимаем синюю кнопку Commit, тем самым создаем новый коммит:

Рисунок 3

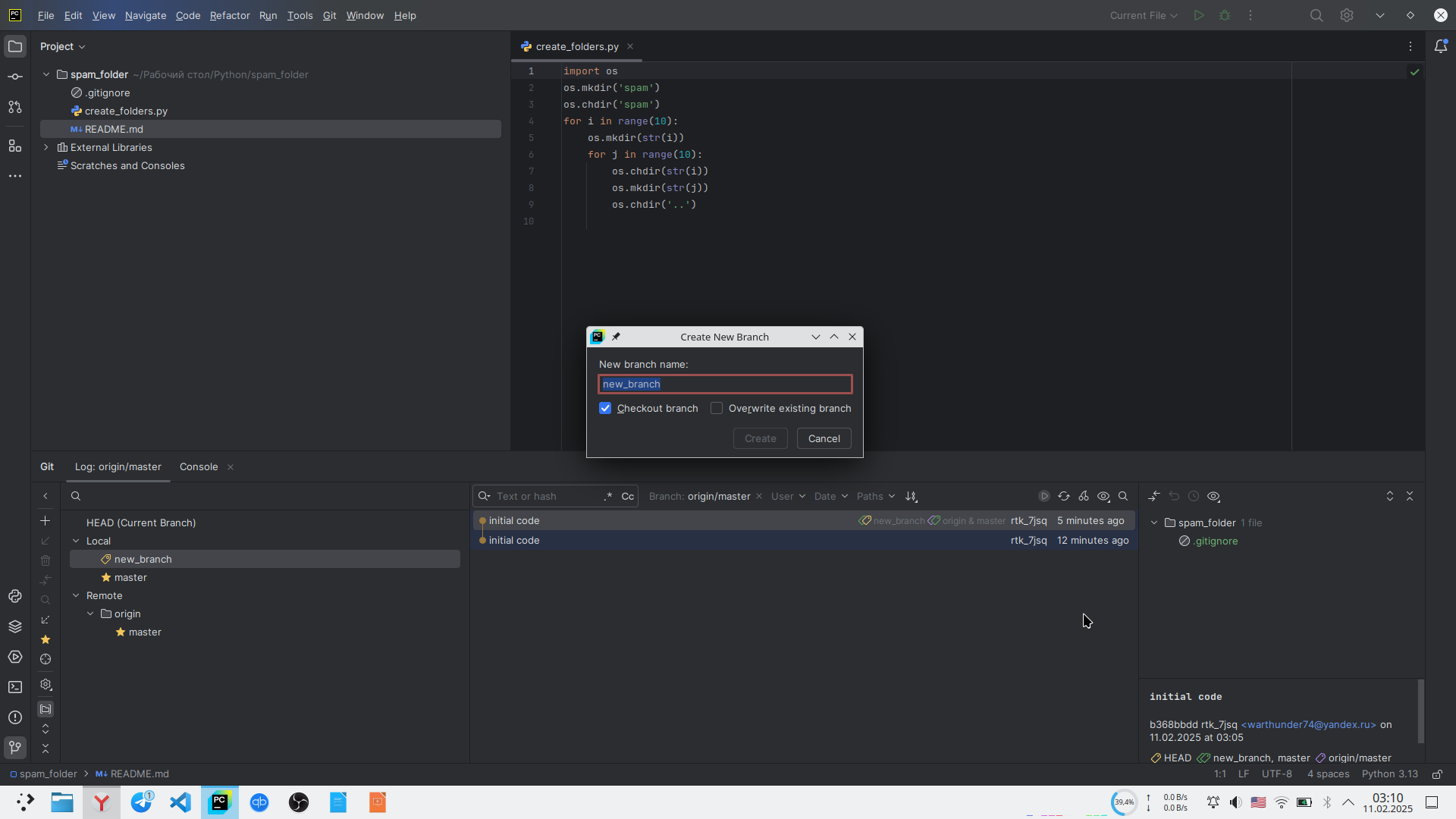
Теперь выгрузим наш репозиторий на GitHub, для этого переходим в меню Git > Manage Remotes и вставляем ссылку нашего удаленного репозитория GitHub, предварительно зарегистрировавшись на нём:

Рисунок 4

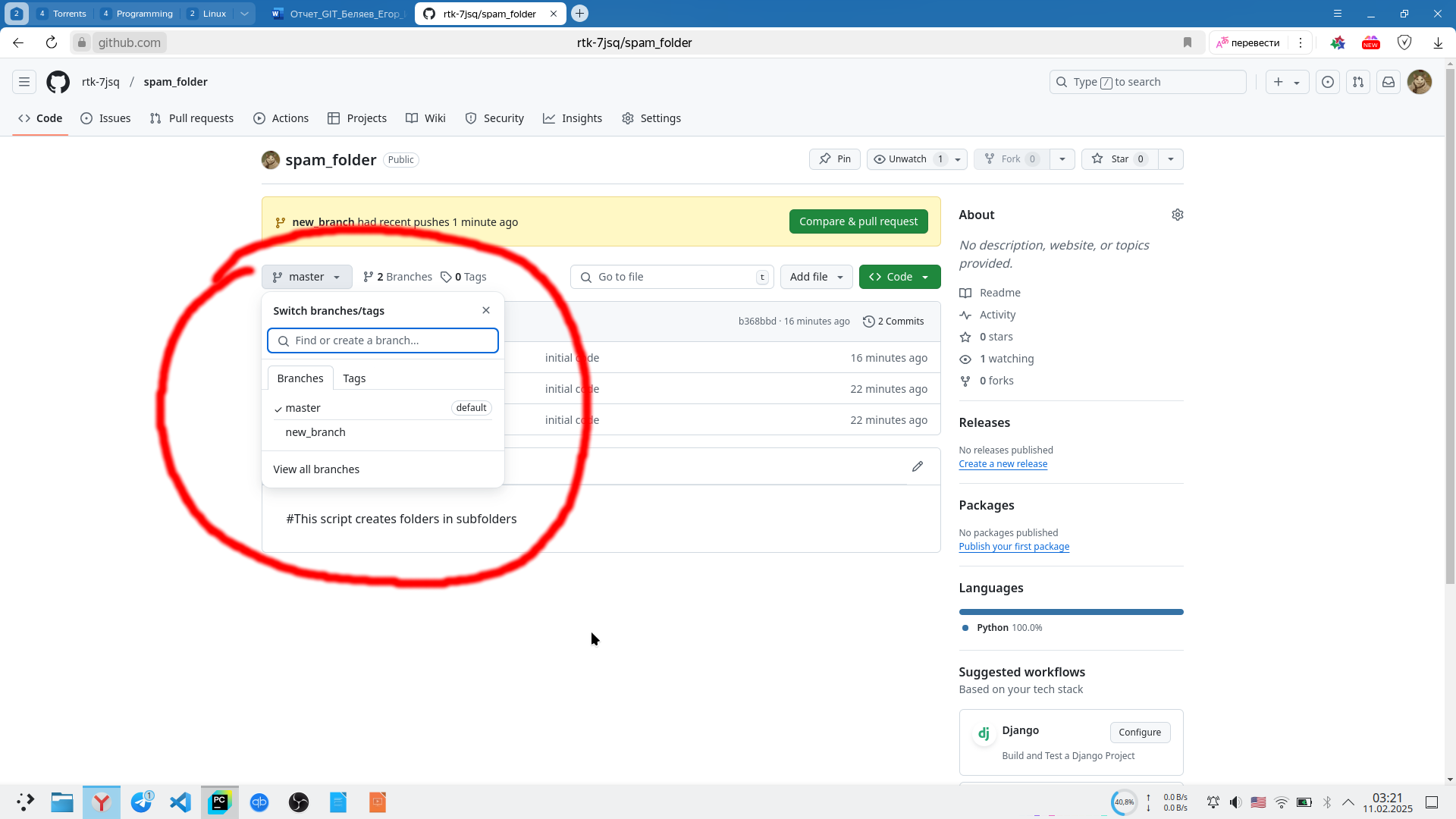
После добавления ссылки возвращаемся в меню Commit и нажимаем Commit and Push, выбираем нужный коммит и отправляем, попутно авторизируем наш компьютер на нашем репозитории, в профиле GitHub получаем наш репозиторий:

Рисунок 5

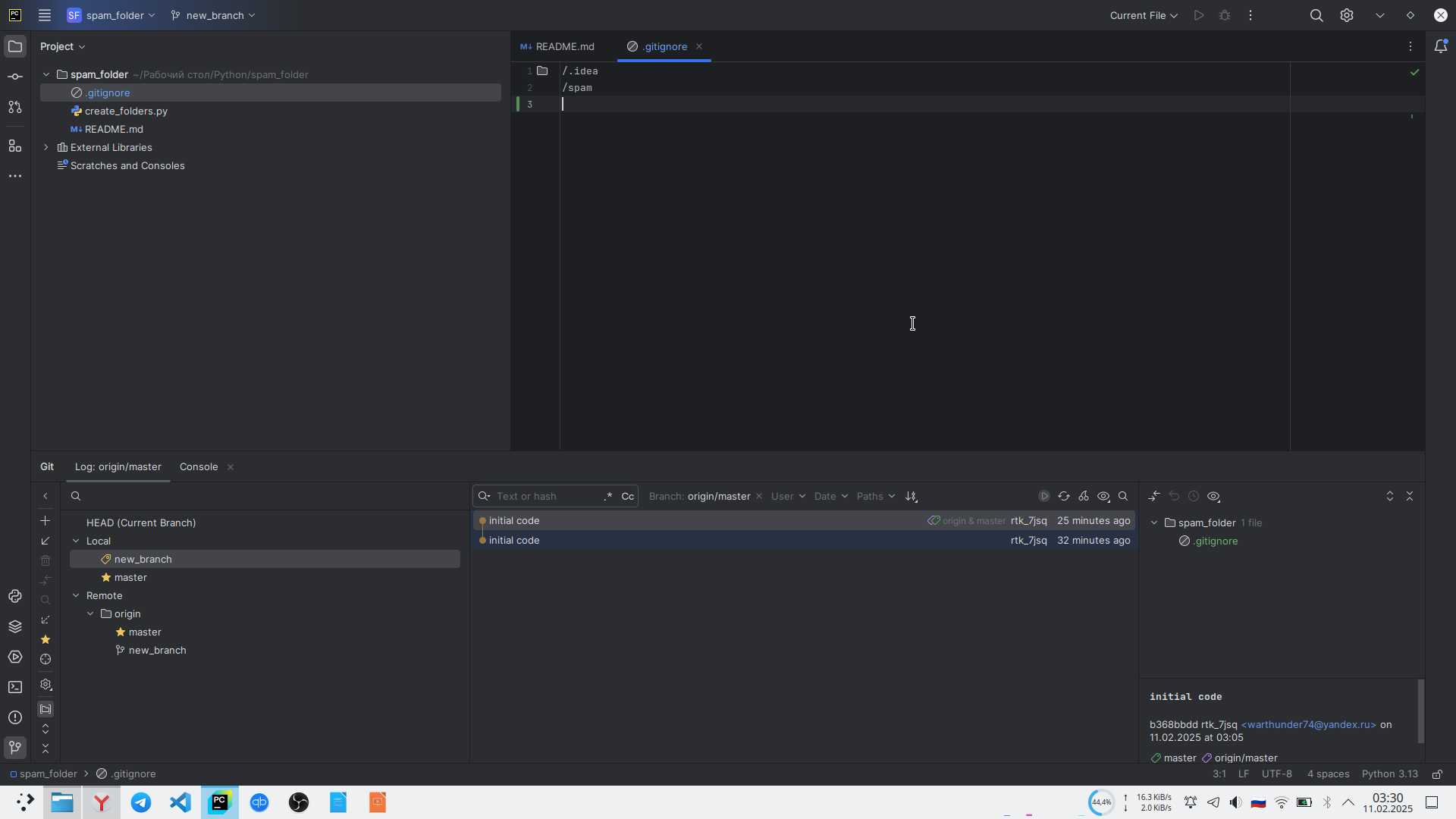
Теперь создадим новую ветку для изменений, для этого переходим в меню Git > New branch и вводим любое имя:

Рисунок 6

Мы сразу же переключимся на нашу новую ветку. Теперь изменим любой файл, например README.md и создадим новый коммит и отправив его на GitHub. После этого на удаленном репозитории появится вторая ветка с нашим коммитом:

Рисунок 7

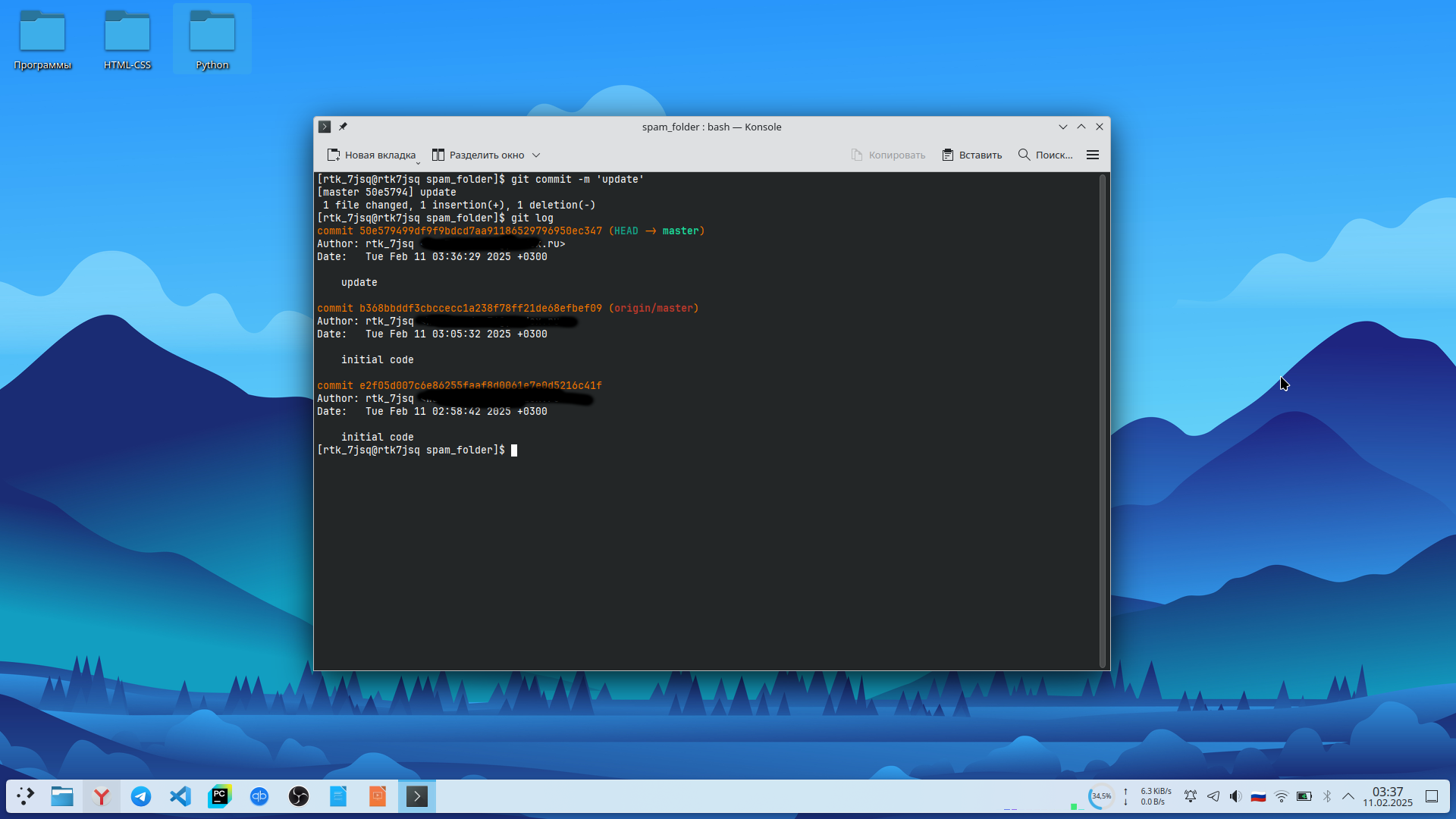
Создадим файл .gitignore для игнорирования ненужных папок при отслеживании, впишем туда папки /.idea и /spam. Вторая папка будет появляться в директории после выполнения кода, поэтому она не нужна нам в удаленном репозитории:

Рисунок 8

Всё тоже самое можно делать из под терминала и даже больше, например создадим коммит, для этого потребуется открыть терминал в папке нашего проекта и ввести следующую команду:

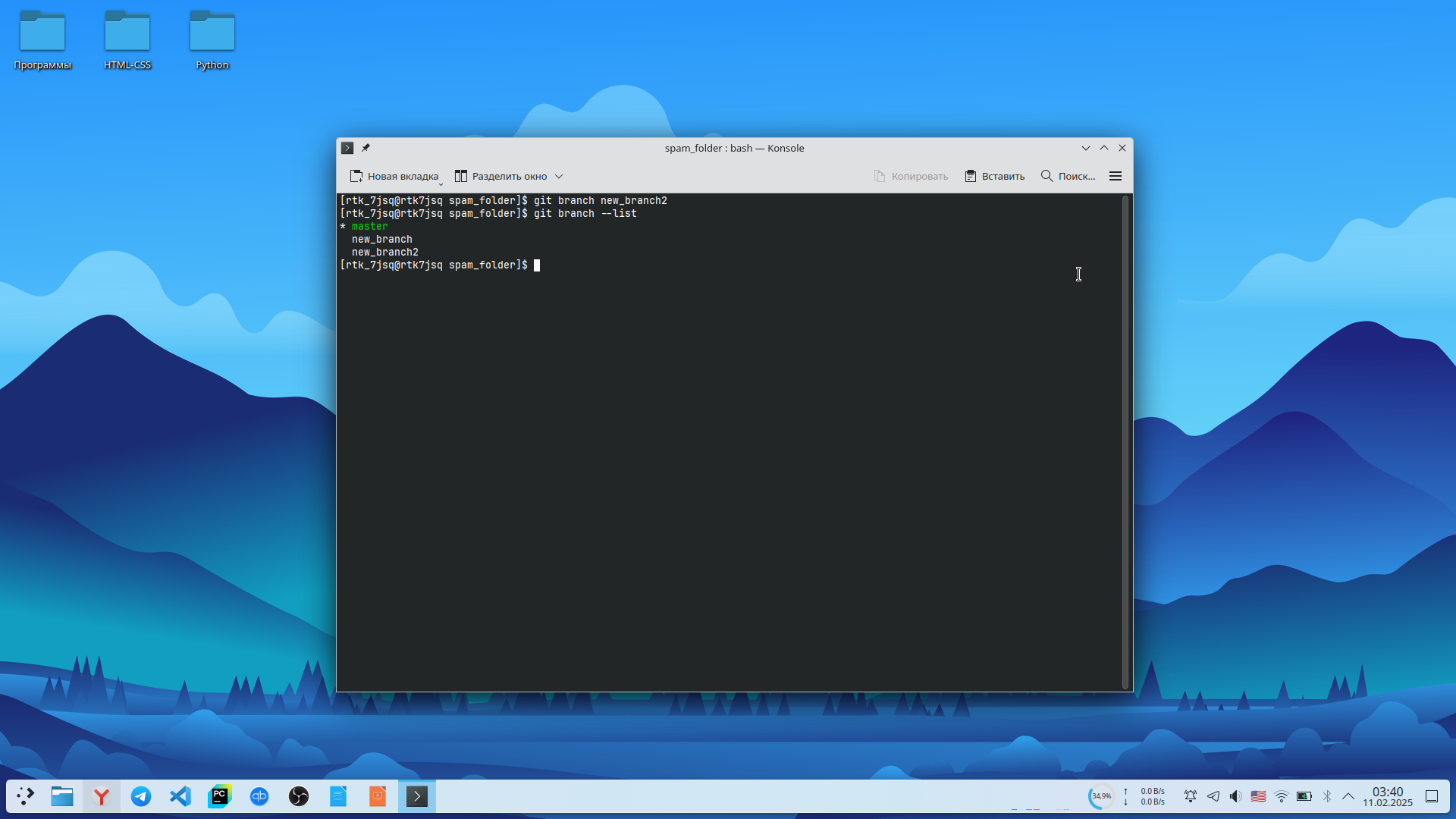
**git commit –m ‘update’**

А после выведем все текущие коммиты командой **git log:**

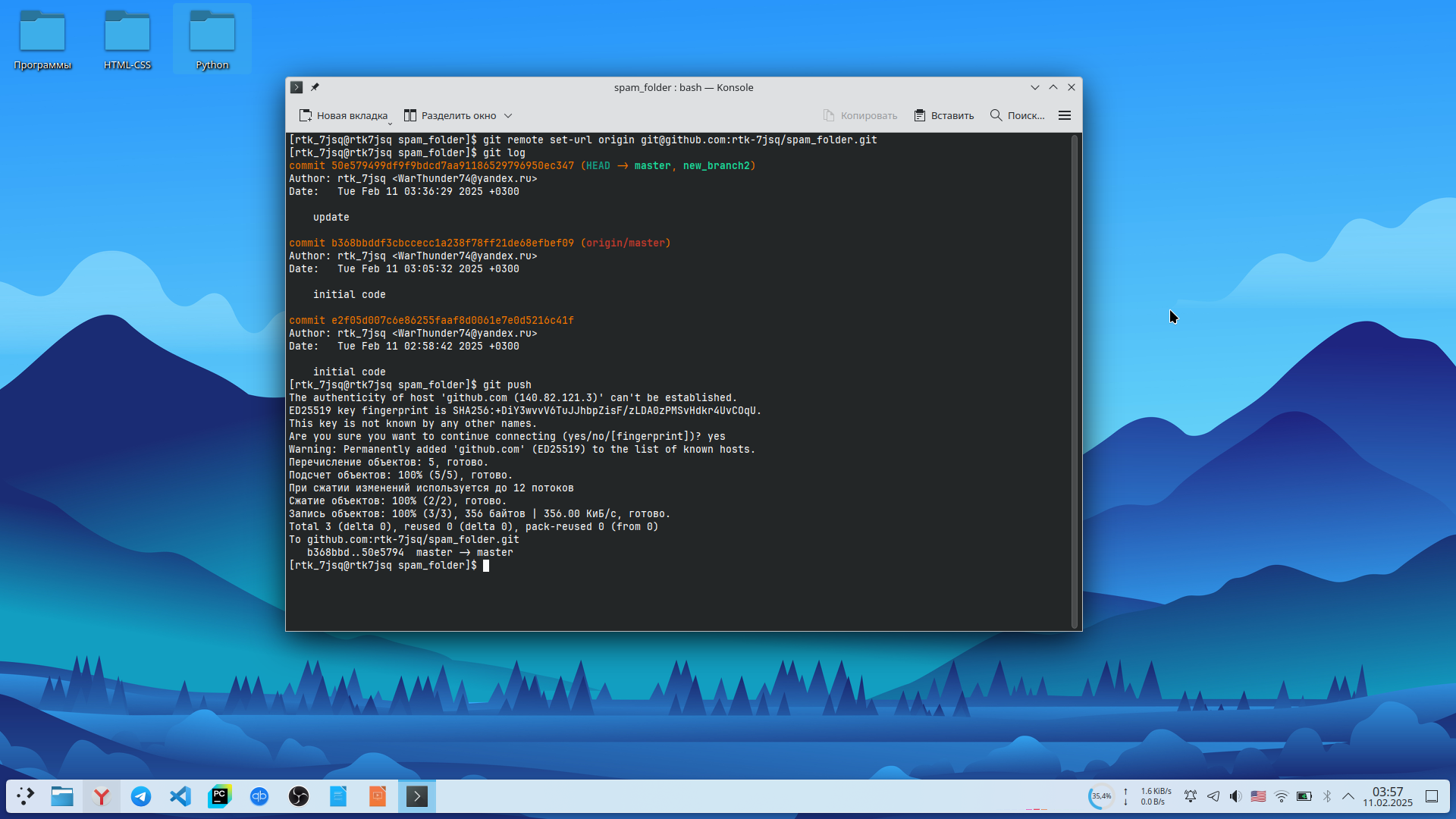
Рисунок 9

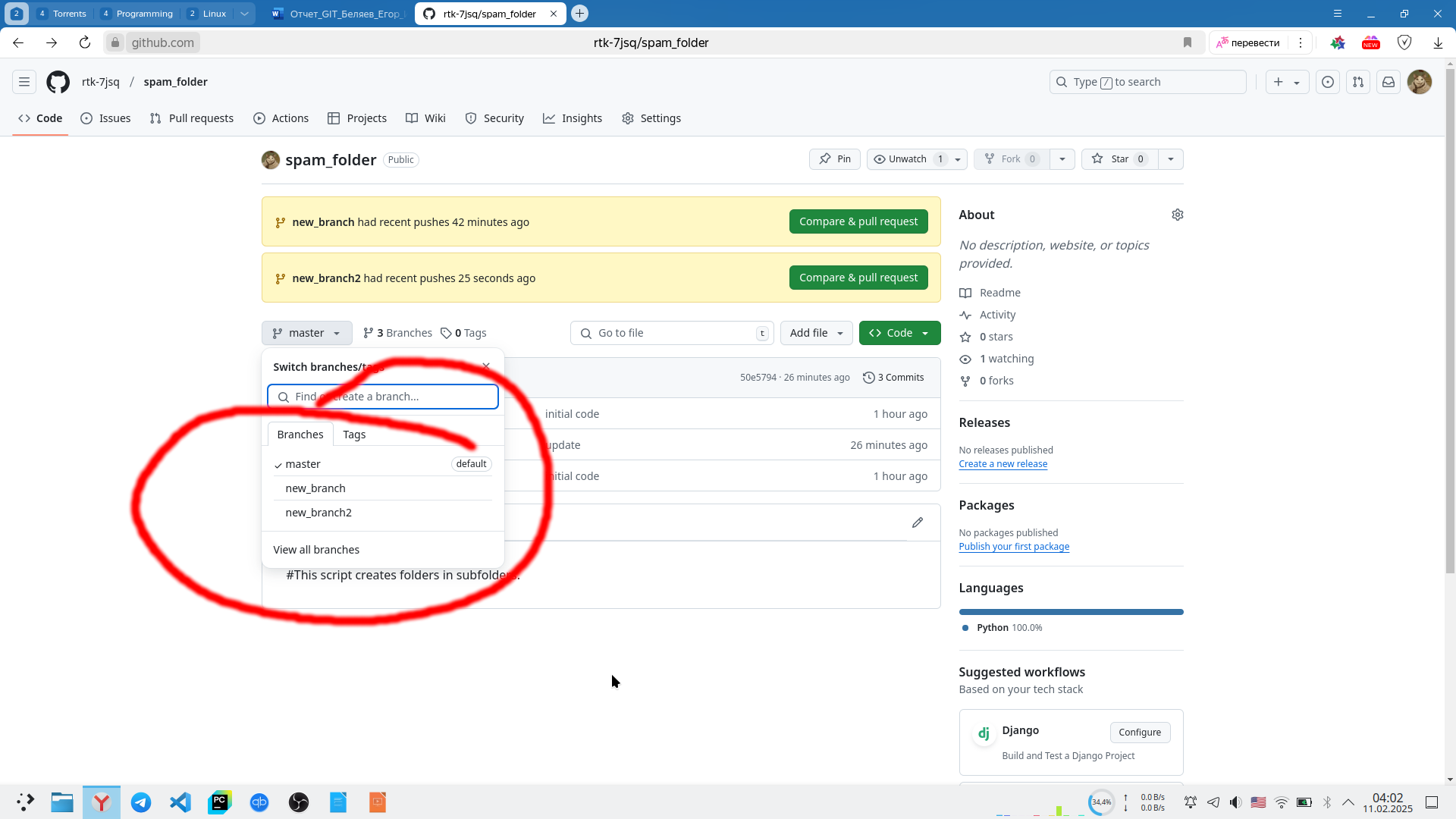
Наш новый коммит находится на первом месте.

Для создания новой ветки можно использовать команду **git branch new\_branch2,** а для вывода списка всех веток команду **git branch –list:**

Рисунок 10

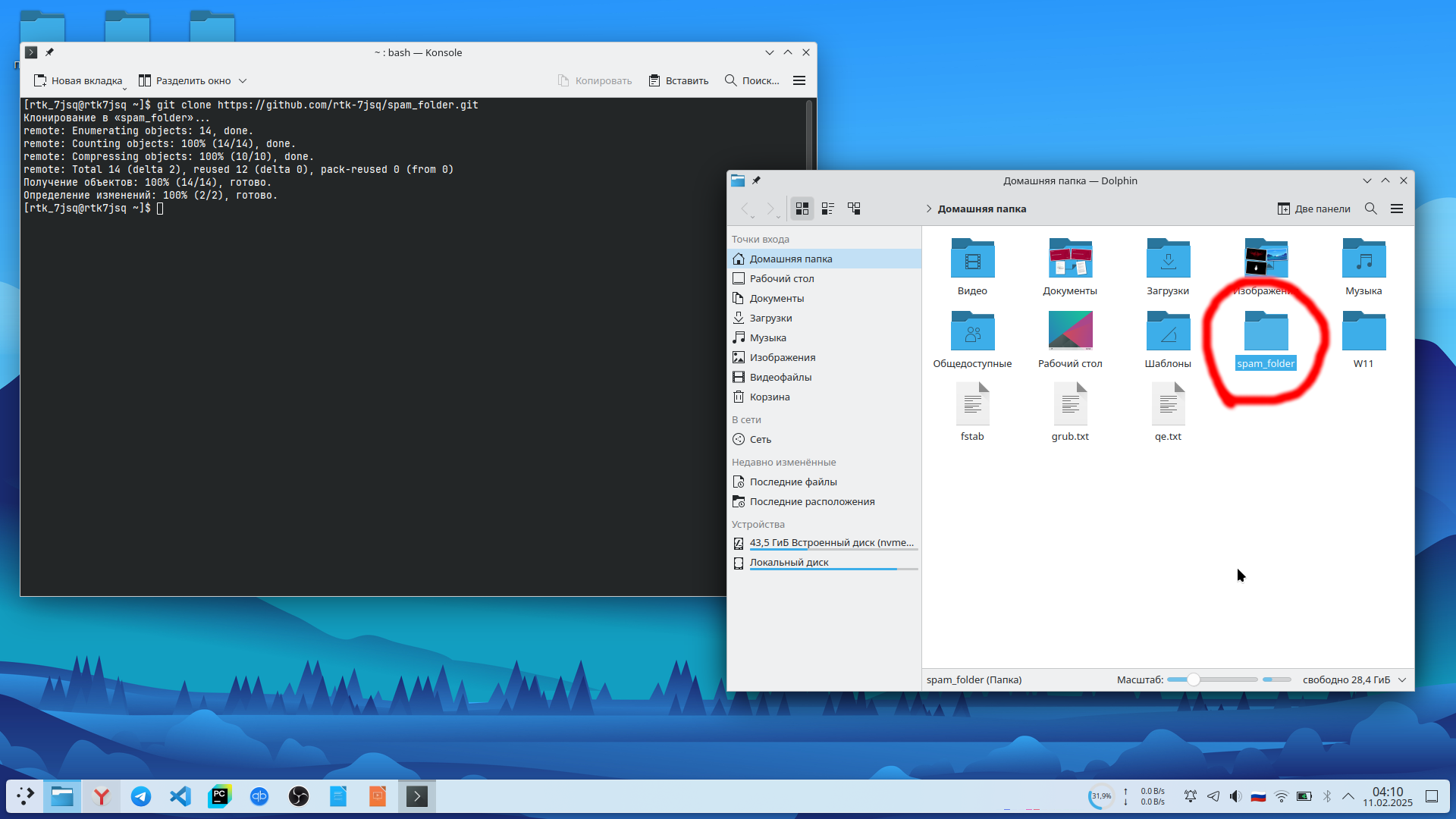
Для итогового отправления коммитов в удаленный репозиторий GitHub используем команду **git push,** после чего проверяем нашу новую ветку и коммит в браузере:

Рисунок 11

Рисунок 12

Также, чтобы конечному пользователю скачать наш репозиторий, необходимо скопировать URL-ссылку нашего репозитория и вставить её в следующую команду:

**git clone** [**https://github.com/rtk-7jsq/spam\_folder.git**](https://github.com/rtk-7jsq/spam_folder.git)

Рисунок 13

Основной список команд и их назначение:

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| git init | Инициализация нового репозитория Git. |
| git clone [url] | Клонирование существующего репозитория из удаленного источника. |
| git add [файлы] | Добавление изменений в индекс (стейджинг). |
| git commit -m "сообщение" | Создание коммита с сохранением изменений и добавлением сообщения. |
| git status | Просмотр статуса репозитория (изменения, что на стадии коммита). |
| git pull | Получение и объединение изменений из удаленного репозитория. |
| git push | Отправка локальных изменений в удаленный репозиторий. |
| git branch | Просмотр веток в репозитории и текущей ветки. |
| git checkout [ветка] | Переключение между ветками или восстановление файлов. |
| git merge [ветка] | Объединение изменений из одной ветки в другую. |
| git log | Просмотр истории коммитов. |

# **Вывод**

Познакомился с такими системами как Git, GitHub, GitLab, GitVerse, а также научился работать с Git репозиториями и изучил основные команды для работы с Git.