МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
(ФГБОУ ВО «ВГТУ», ВГТУ)

Факультет информационных технологий и компьютерной безопасности

Кафедра компьютерных интеллектуальных технологий проектирования

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

По дисциплине: «Системы хранения и обработки данных»

Тема: «Основы работы с системой управления версиями Git»

Выполнил работу студент группы змИИВТ-241: Авсянкина М.А.

подпись, дата

Принял: Короленко В.В.

подпись, дата

Воронеж 2024

Цель работы: изучить назначение распределённой системы управления версиями, освоить процесс создания репозитория и основы управления версиями файлов.

Основные задачи:

• создание профиля и репозитория на платформе github;

• установка git, запуск;

• создание ветки для отработки своего проекта;

• изучение основных команд git;

• клонирование удаленного репозитория на локальный компьютер;

• создание файла с данными о себе, изменение файла и отправка соответствующих изменений из локального репозитория на удаленный;

• откат изменений до предыдущего коммита.

Ход работы:

В данной лабораторной работе мы познакомимся с платформой githud.

GitHub — это веб-сервис для хостинга IT-проектов и их совместной разработки. Он предоставляет платформу для хранения исходного кода программ, управления версиями и совместной работы над проектами. GitHub используется разработчиками для публикации своих проектов, внесения изменений, отслеживания версий и взаимодействия с другими участниками сообщества. Также сервис предлагает инструменты для создания вики-страниц, ведения блогов и выпуска релизов.

Мы начнем с основ: создание и настройка первого репозитория, управление ветками, внесение изменений и их фиксация, а также работа с удаленными репозиториями. Мы также рассмотрим, как использовать Git для совместной работы с командой, проводить код-ревью и публиковать ваши проекты на платформах вроде GitHub.

Для начала, создадим профиль на данной платформе (рисунок 1).

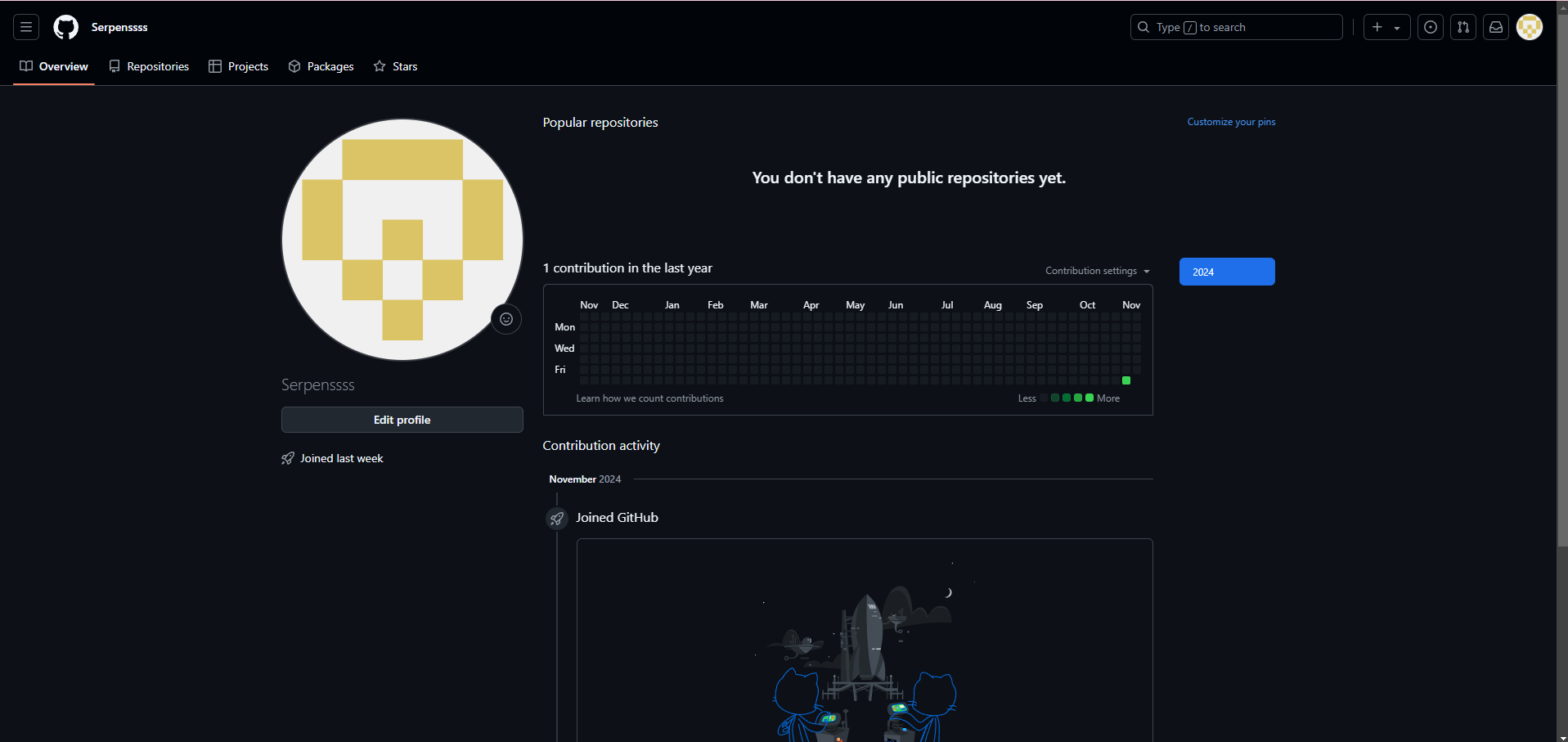


Рисунок 1 – Профиль на платформе GitHub

Далее создаем репозиторий. Он нужен для хранения данных, где разработчики могут размещать свои проекты, такие как код, файлы, документацию и другие ресурсы, связанные с проектом. Репозитории используются для управления версиями проекта, отслеживания изменений, координации работы команды разработчиков и обеспечения возможности обратной связи и вкладов от сообщества. Каждый проект на GitHub имеет свой уникальный репозиторий, который можно создавать, управлять и делиться им с другими пользователями платформы.

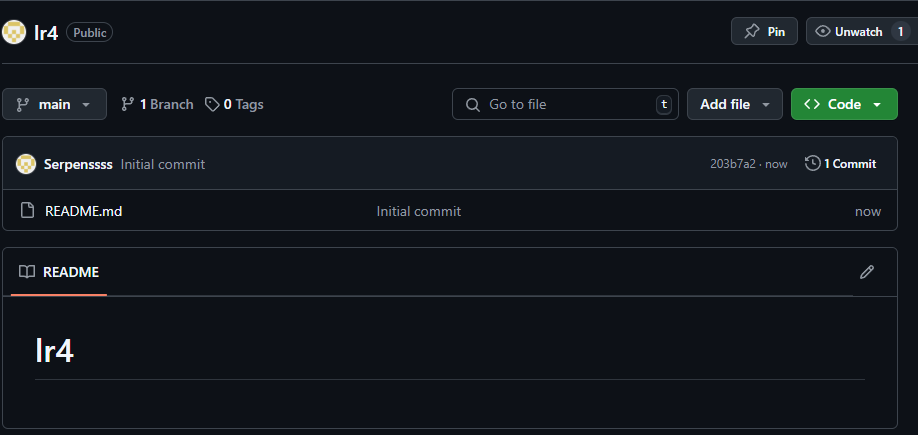


Рисунок 2 – Созданный репозиторий LR4

Указываем в файле readme.md данного репозитория информацию, для чего он был создан и непосредственно кем (рисунок 3).

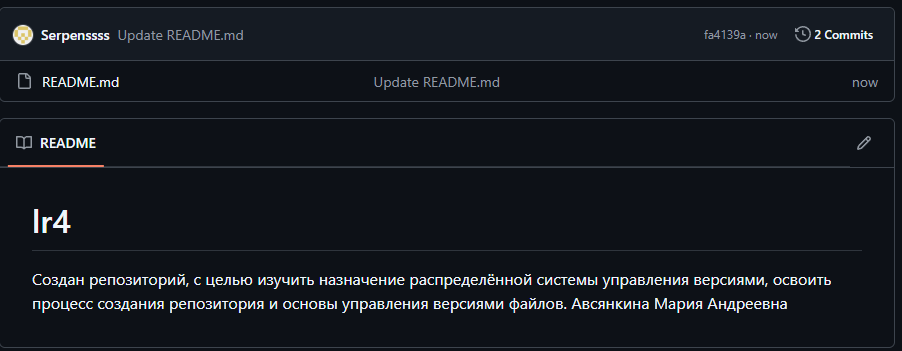


Рисунок 3 – Комментарий в файле readme.md

Теперь делаем ответвление от главной ветки (main). Ответвления позволяют поддерживать чистоту и организованность кодовой базы, обеспечивают возможность параллельной работы над различными направлениями развития проекта и помогают управлять сложными процессами разработки. Создаем новую ветку с названием LR4-AMA (рисунок 4).

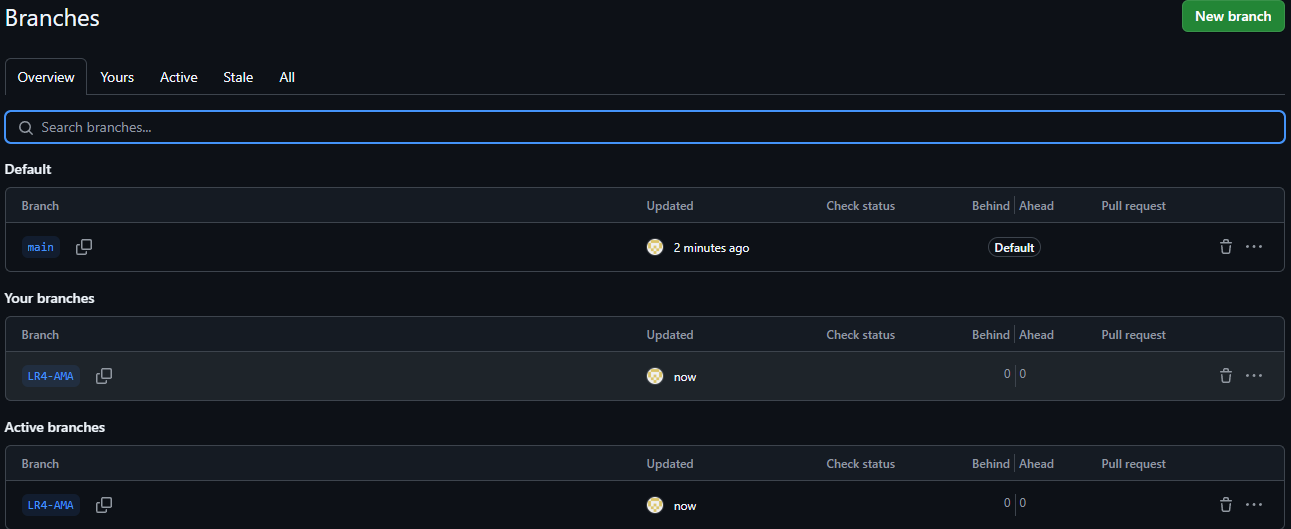


Рисунок 4 – Новая ветка LR4-AMA

Закончив подготовку для создания нашего нового проекта, и работы в нем, приступаем к изучению основных команд git.

Основные команды Git и их назначение:

1. git init: Инициализация нового репозитория. Команда создает начальный пул (корневой каталог) для хранения всех данных репозитория.
2. git clone: Копирование существующего репозитория. Эта команда создает локальную копию удаленного репозитория, включая историю коммитов и рабочие файлы.
3. git add: Добавление файлов в индекс. После того как файлы были изменены, они добавляются в индекс перед коммитом.
4. git commit: Фиксация изменений. Команда сохраняет текущее состояние рабочей директории в истории коммитов. Обязательно нужно указать комментарий (message), объясняющий изменения.
5. git log: Просмотр истории коммитов. Позволяет увидеть список всех предыдущих коммитов, их даты, авторов и сообщения.
6. git diff: Сравнение изменений. Эта команда показывает различия между текущим состоянием рабочего дерева и последним коммитом или другим выбранным коммитом.
7. git status: Проверка состояния. Выводит информацию о текущем состоянии репозитория, включая какие файлы были изменены, добавлены или удалены.
8. git branch: Управление ветками. Команда позволяет создавать, удалять, переключаться между ветками и просматривать список активных веток.
9. git merge: Объединение веток. Соединяет две ветки вместе, объединяя их изменения.
10. git push: Публикация изменений. Отправляет локальные коммиты на удаленный сервер (например, GitHub).
11. git pull: Загрузка изменений. Скачивает изменения с удаленного сервера и интегрирует их в локальный репозиторий.
12. git done: Клонирует удаленный репозиторий по указанному URL-адресу и создает его локальную копию
13. git config: Устанавливает ваше имя как автора коммитов.

Эти команды являются основой для работы с Git и позволяют эффективно управлять историей разработки, работать с ветками, синхронизировать изменения между различными репозиториями и контролировать процесс разработки.

Внесем данную информацию в ветку LR4-AMA, файл readme.md (рисунок 5).

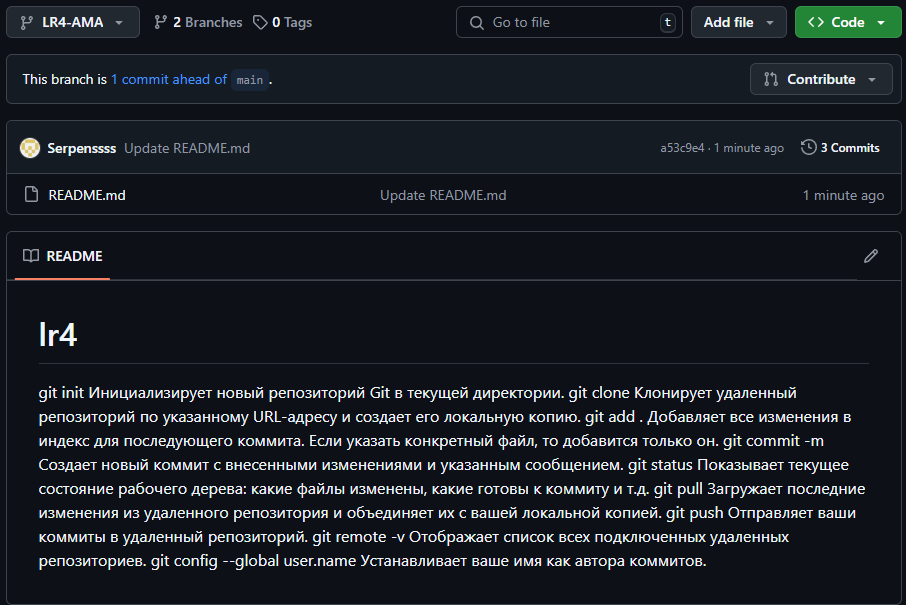


Рисунок 5 – Основные команды Git в ветке LR4-AMA

Для начала работы на локальном компьютере с платформой GitHub, настраиваем подключение по SSH (рисунок 6).

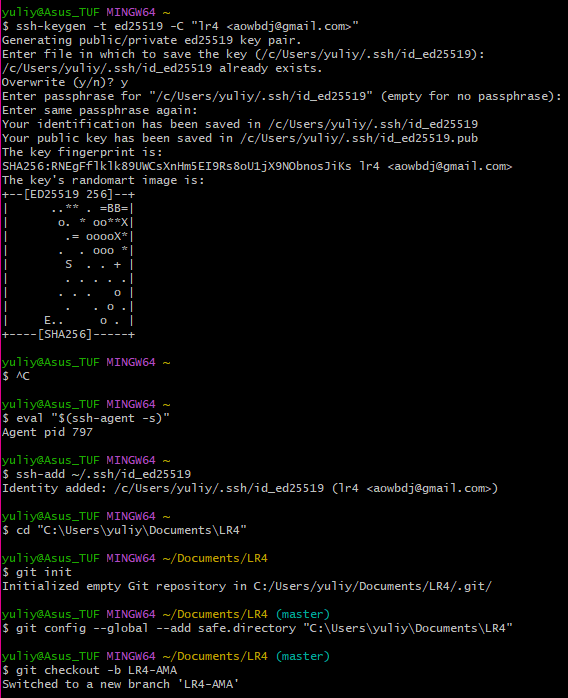


Рисунок 6 – Настройка подключения по SSH

Согласно рисунку 6, сначала мы создаем SSH ключ, который будет указывать на того, что он означает, с помощью команды:

$ ssh-keygen -t ed25519 -C "lr4 aowbdj@gmail.com"

После добавляем ключ в SSH-агента командой:

$ eval "$(ssh-agent -s)"

И с помощью команды ниже, указывает путь до приватного ключа:

$ ssh-add ~/.ssh/id\_ed25519

Теперь мы можем зайти, с помощью терминала в созданную для проекта папку. Потом инициализируем созданный репозиторий. Как видно из рисунка 6, после команды:

$ git init

Мы заходим в наш репозиторий, в ветку master (main). Таким образом, мы активировали локальный репозиторий и связали его с локальным компьютером, из активной папки.

Дальнейшую работу будем выполнять непосредственно из нашей личной ветки (LR4-AMA). Теперь мы начинаем работать с файловыми данными. Создаем в локальном репозитории текстовый файл, с названием AMA.txt, с данными о себе (рисунок 7).

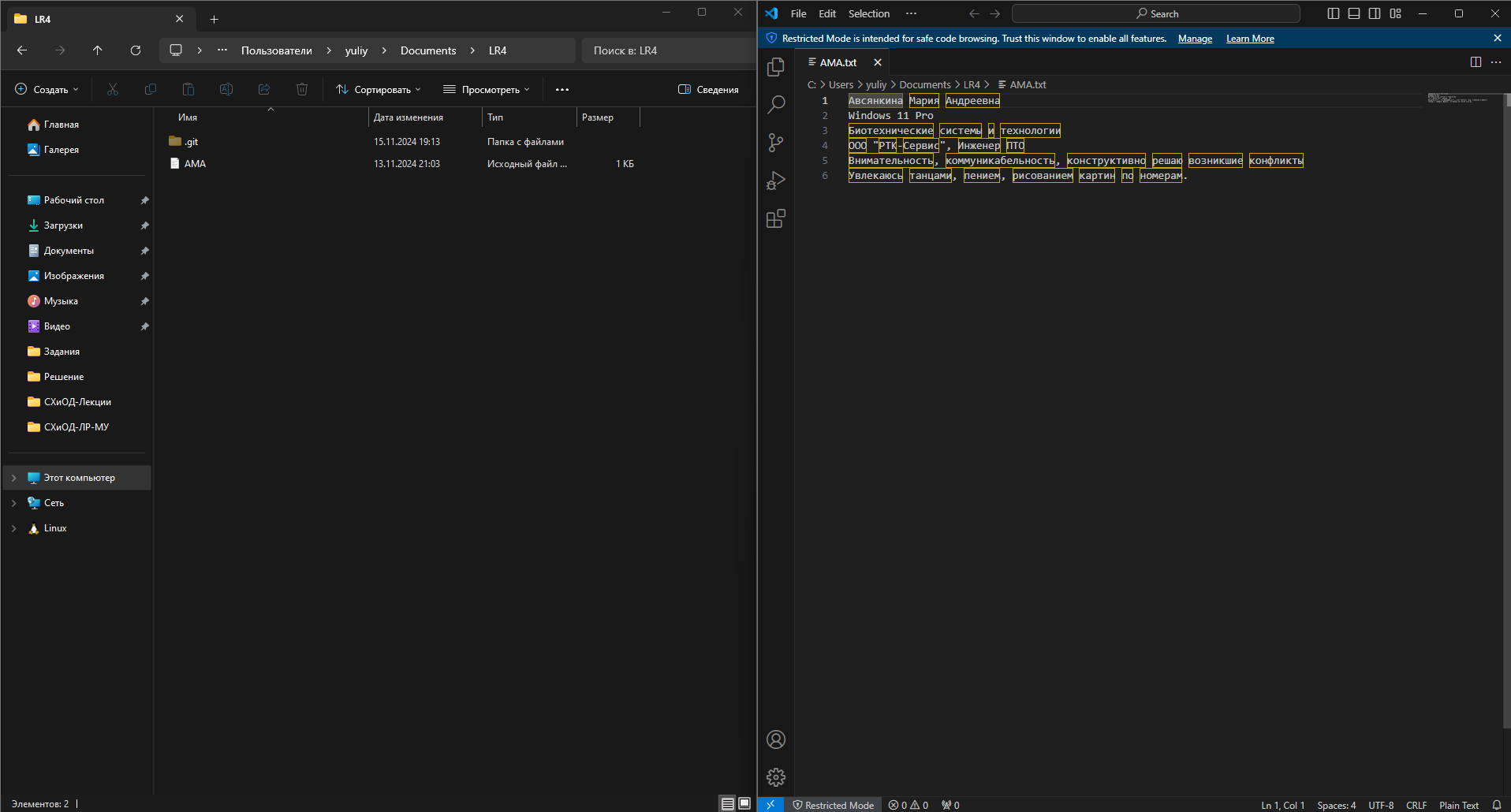


Рисунок 7 – Файл AMA.txt с данными о себе

Сохраняем созданные изменения, и копируем созданный локально файл на удаленный репозиторий в ветку LR4-AMA. Для этого, сначала зайдем в нашу ветку на терминале, с помощью команды:

$ git checkout -b LR4-AMA

На рисунке 8, во второй строчке отображается, что мы как раз зашли в нашу ранее созданную ветку. Проверяем статус текущего репозитория git. Статус указывает на то, что мы находимся в ветке LR4-AMA, в ней отсутствуют коммиты и файл AMA.txt находится в статусе неотслеженный. Он находится в нашей локальной ветке, но ещё не был добавлен в индекс Git.

Подготавливаем внесенные изменения к отправке, командой:

$ git add .

После чего делаем коммит в Git об отправленном созданном файле локально, на удаленным репозиторий.

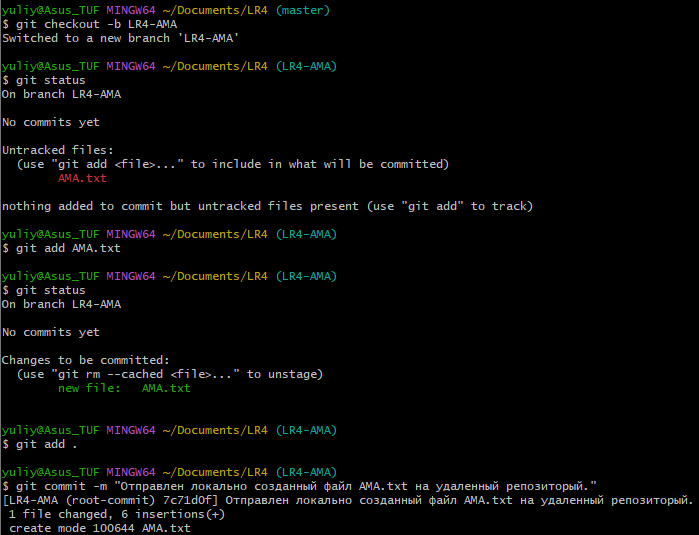


Рисунок 8 – Копирование созданного локального репозитория на удалённый в ветку LR4-AMA

Настраиваем дополнительно связь с удаленным репозиторием (рисунок 9)



Рисунок 9 – Настройка связи с удаленным репозиторием

И непосредственно делаем синхронизацию данных, согласно рисунку 10.

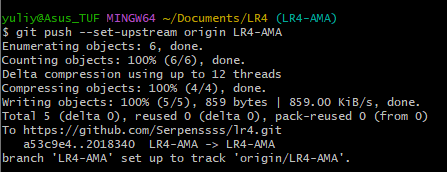


Рисунок 10 – Назначение ветки синхронизации LR4-AMA

Проконтролируем, отобразились ли наши внесенные изменения, на платформе GitHub (рисунок 11).

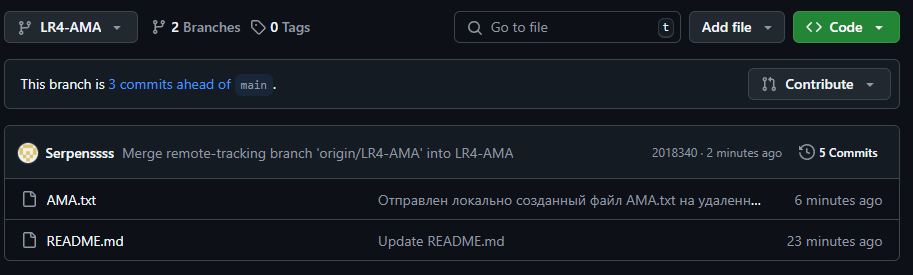


Рисунок 11 – Добавление файла AMA.txt на платформу git

Проверим, корректно ли перенеслись наши внесенные данные в файл AMA.txt (рисунок 12).

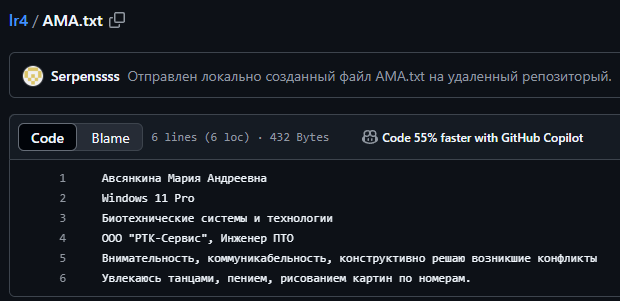


Рисунок 12 – Отображение данных файла AMA.txt на платформе git

Как можно заметить, данный файл находится в ветке LR4-AMA, что полностью подтверждает правильность выполненных наших действий ранее.

Далее научимся отправлять изменения с локального репозитория на удаленный. Открываем созданный ранее файл AMA.txt и добавляем в него данные о среднем балле по предыдущему образованию (рисунок 13).

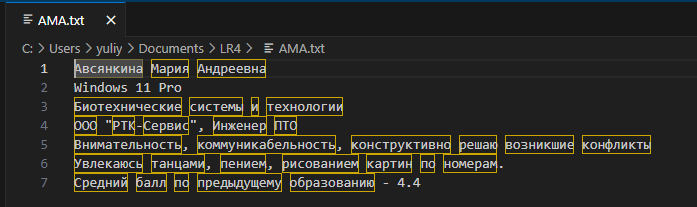


Рисунок 13 – Добавление данных о среднем балле в файл AMA.txt

Проверяем статус файла, видим, наш файл был модифицирован в локальном репозитории, но еще не добавлен в удаленный. Подготавливаем изменения к отправке, оставляем коммит в githud и отправляем внесенные изменения в нужную нам ветку LR4-AMA.

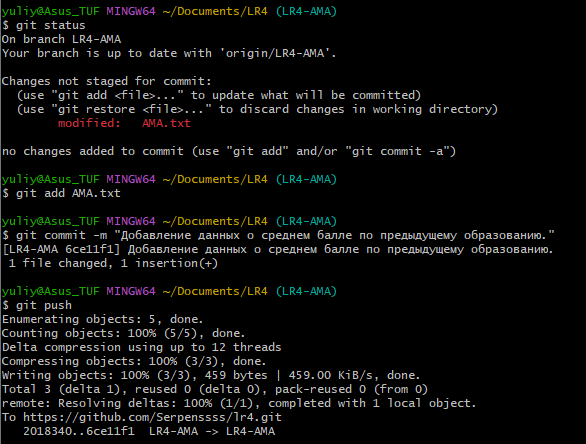


Рисунок 14 – Отправка изменений после добавления данных о среднем балле в файле AMA.txt

Проверим, действительно ли внесенные изменения на локальном репозитории появились на удаленном.

Открывая коммиты, видим, что появился новый в нашей используемой ветке (рисунок 15).

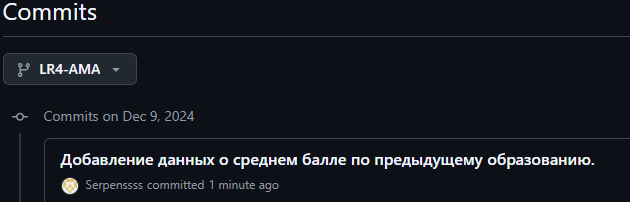


Рисунок 15 – Коммит о “Добавление данных о среднем балле по предыдущему образованию”

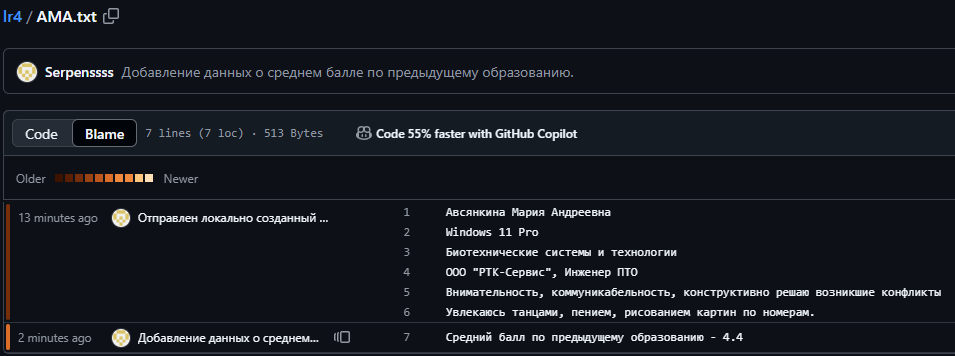


Рисунок 16 – Отображение обновления в файле AMA.txt

Повторим вышеперечисленные действия для добавления новых данных о месте рождение.

Добавляем локально информацию о месте рождения (рисунок 17).

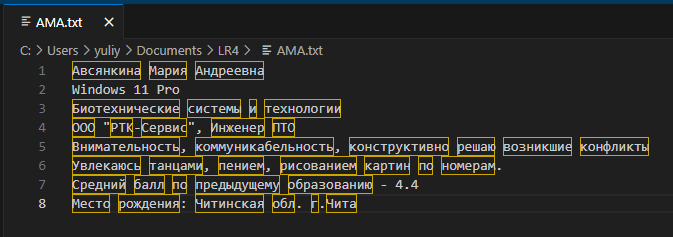


Рисунок 17 – Изменения в файле AMA.txt локально

Проверяем статус, обновляем данные, оставляем коммит и отправляем данные на удаленный репозиторий (рисунок 18).

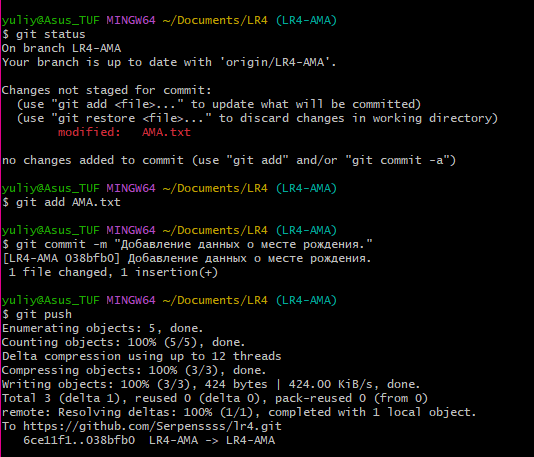


Рисунок 18 – Отправка данных на удаленный репозиторий о “Добавление данных о месте рождения”

Проверяем фактическое обновление данных на удаленном репозитории (рисунок 19-20).

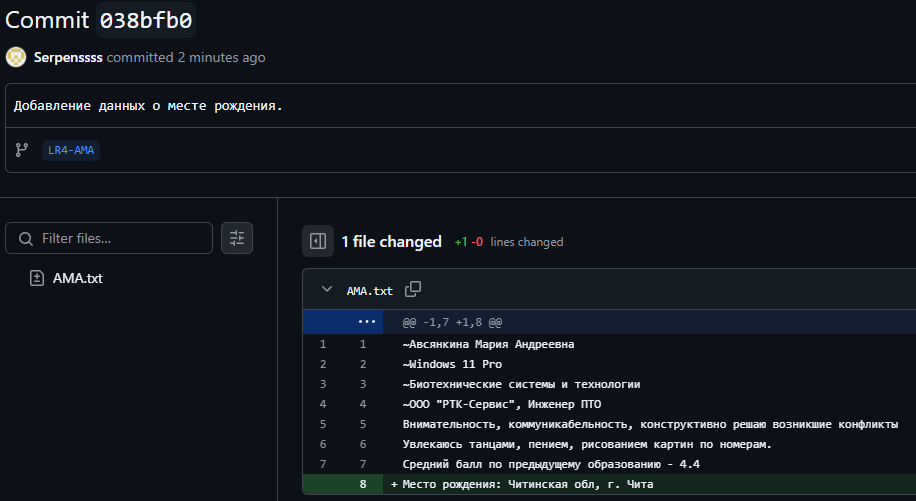


Рисунок 19 – Обновление файла AMA.txt

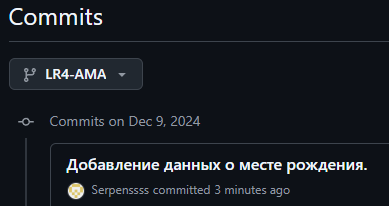


Рисунок 20 – Появился новый коммит “Добавление данных о месте рождения”

Научимся делать откат до предыдущего коммита. Для этого, сначала надо узнать id коммита. Сделать это можно с помощью команды:

$ git log

Находим id нужного коммита, который нам надо откатить, в нашем случае это первый коммит с id = 038bfb0547c0a04b5c8f42b7bfdfdaf1e64fec41. Вводим команду:

$ git revert 038bfb0547c0a04b5c8f42b7bfdfdaf1e64fec41

Подготавливаем изменения к отправке, и добавляем их в удаленный репозиторий (рисунок 21).

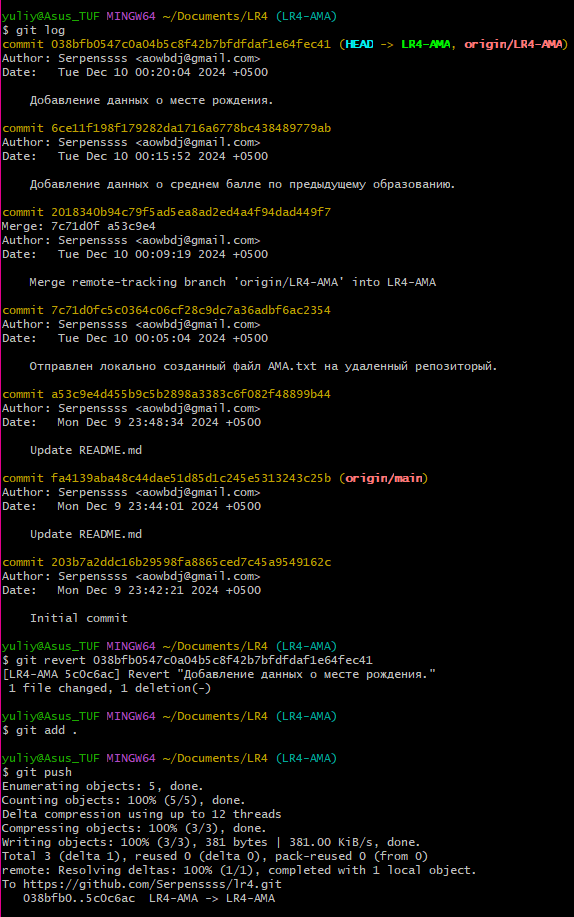


Рисунок 21 – Откат изменений до прошлого коммита

Проконтролируем и посмотрим, какие изменения мы внесли и как они отображаются на платформе git (рисунок 22).

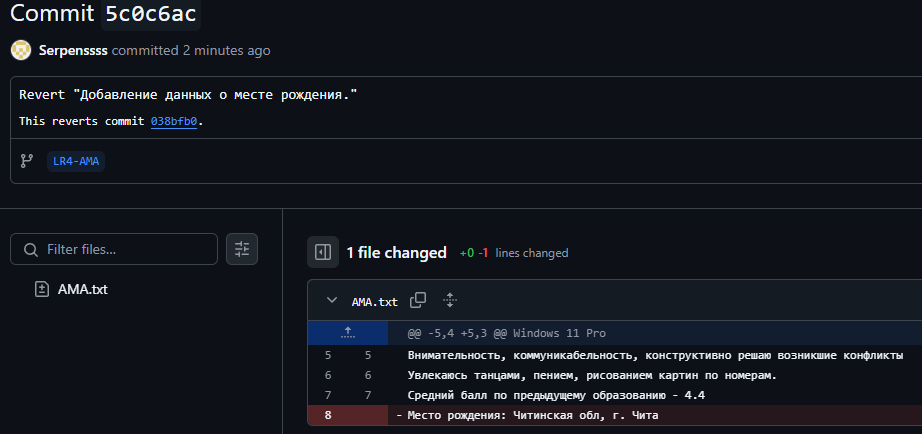


Рисунок 22 – Откат последнего добавленного коммита в файл AMA.txt

Как видно из рисунка 22, 8 строчка сделала откат, а добавление 7 строчки осталось на месте.

Вывод: таким образом, мы изучили назначение распределённой системы управления версиями, освоили процесс создания репозитория и основы управления версиями файлов. Так же изучили основные команды git, научились делать откаты до прошлого коммитов.

***Контрольные вопросы:***

1. Что такое Git?

– Git – это распределённая система контроля версий, которая используется для отслеживания изменений в файлах и координации совместной работы нескольких разработчиков над одним проектом. Удобно работать с разными ветками и объединять изменения различных участников.

1. Для чего команда разработки использует Git?

– Помогает отслеживать изменения в коде, что особенно важно в больших проектах, где работает несколько человек. Упрощает совместную разработку, позволяя разным участникам работать параллельно над разными частями проекта. Снижает риск потери данных, так как Git хранит всю историю изменений.

1. Основные команды и операции, необходимые для работы в Git.

– git init, git clone, git add, git commit -m, git status, git log, git checkout, git branch, git merge, git pull, git push

1. Что значит инициализация репозитория, какая команда для этого используется?

* Инициализация репозитория означает создание нового пустого репозитория Git в выбранной папке. Эта операция создаёт скрытый каталог .git, который содержит всю необходимую информацию для управления версиями файлов в этом проекте.

Команда для инициализации репозитория: git init

Ссылка на репозиторий: https://github.com/Serpenssss/lr4.git