**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2**

*дисциплина: Архитектура компьютера*

Студент: Курушин Георгий

Группа: НКАбд-02-24

**МОСКВА**

2024г.

# Содержание

[1. Цель работы 3](#_Toc178433314)

[2. Задание 4](#_Toc178433315)

[3. Теоретическое введение 5](#_Toc178433316)

[4. Выполнение лабораторной работы 7](#_Toc178433317)

[4.1. Настройка GitHub. 7](#_Toc178433318)

[4.2. Базовая настройка git. 7](#_Toc178433319)

[4.3. Создание SSH ключа 8](#_Toc178433320)

[4.4. Создание рабочего пространства, репозитория курса на основе шаблона. 10](#_Toc178433321)

[4.5. Создание репозитория курса на основе шаблона. 10](#_Toc178433322)

[4.6. Настройка каталога курса. 12](#_Toc178433323)

[4.7. Выполнение заданий для самостоятельной работы. 13](#_Toc178433324)

[5. Заключение 16](#_Toc178433325)

[6. Список используемой литературы 17](#_Toc178433326)

# Цель работы

Целью работы является применение средств контроля версий. А также очень важно приобрести практические навыки по работе с системой git.

# Задание

1. Настройка GitHub.
2. Базовая настройка Git.
3. Создание SSH-ключа.
4. Создание рабочего пространства и репозитория курса на основе шаблона.
5. Создание репозитория курса на основе шаблона.
6. Настройка каталога курса.
7. Выполнение заданий для самостоятельной работы.

# Теоретическое введение

Система контроля версий ( Version Control System, VCS ) — это инструмент, используемый разработчиками программного обеспечения для управления изменениями в исходном коде и других файловых ресурсах.

Системы контроля версий разработаны специально для того, чтобы максимально упростить и упорядочить работу над проектом (вне зависимости от того, сколько человек в этом участвуют). СКВ дает возможность видеть, кто, когда и какие изменения вносил; позволяет формировать новые ветви проекта, объединять уже имеющиеся; настраивать контроль доступа к проекту ; осуществлять откат до предыдущих версий.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом. Демидова А. В. 14 Архитектура ЭВМ Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности.

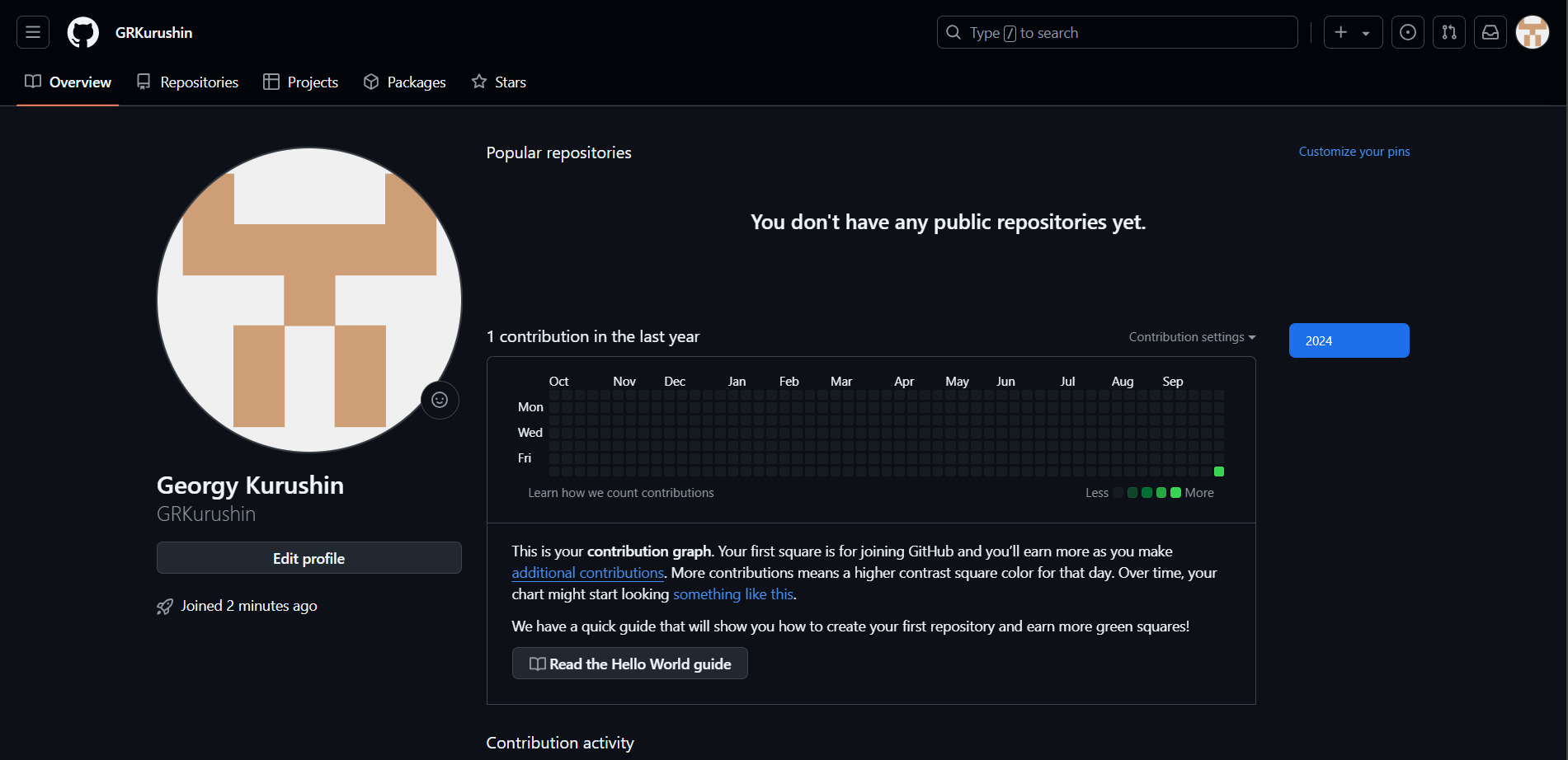
Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным. Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями. Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.

# Выполнение лабораторной работы

## Настройка GitHub.

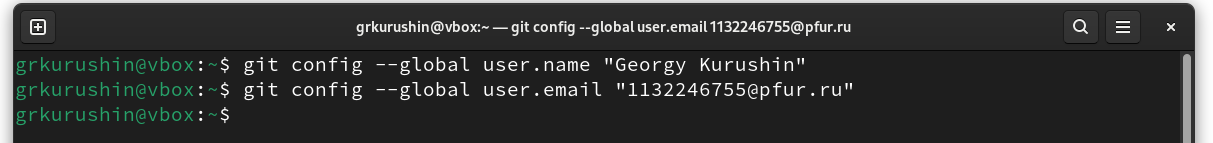
Для выполнения лабораторной работы создаю учетную запись на <https://github.com/> (рис. 4.1.1)



*рис.4.1.1. Учетная запись на GitHub.*

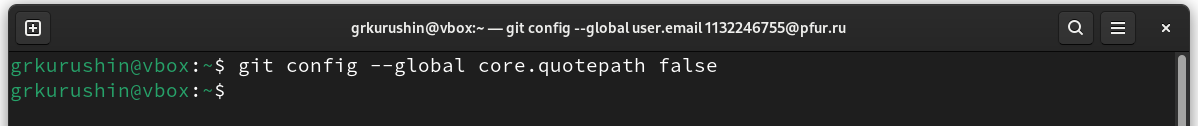
## Базовая настройка git.

Делаю предварительную конфигурацию git. Захожу в терминал и ввожу команды, указывая свое имя и email (рис.4.2.1)



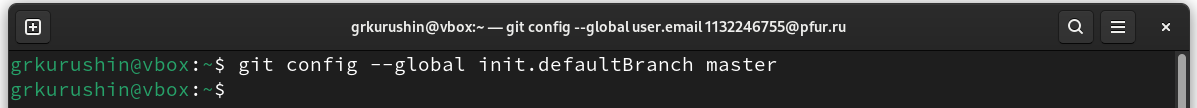
*рис.4.2.1. Предварительная конфигурация в git.*

Настраиваю utf-8 в выходе сообщений git (рис. 4.2.2)



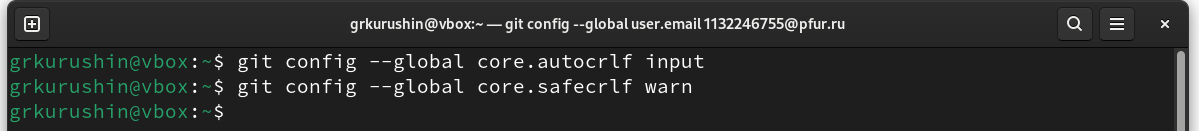
*рис.4.2.2. Настраивание utf-8.*

Задаю имя начальной ветки, которую буду называть master (рис. 4.2.3)



*рис.4.2.3. Имя начальной ветки*

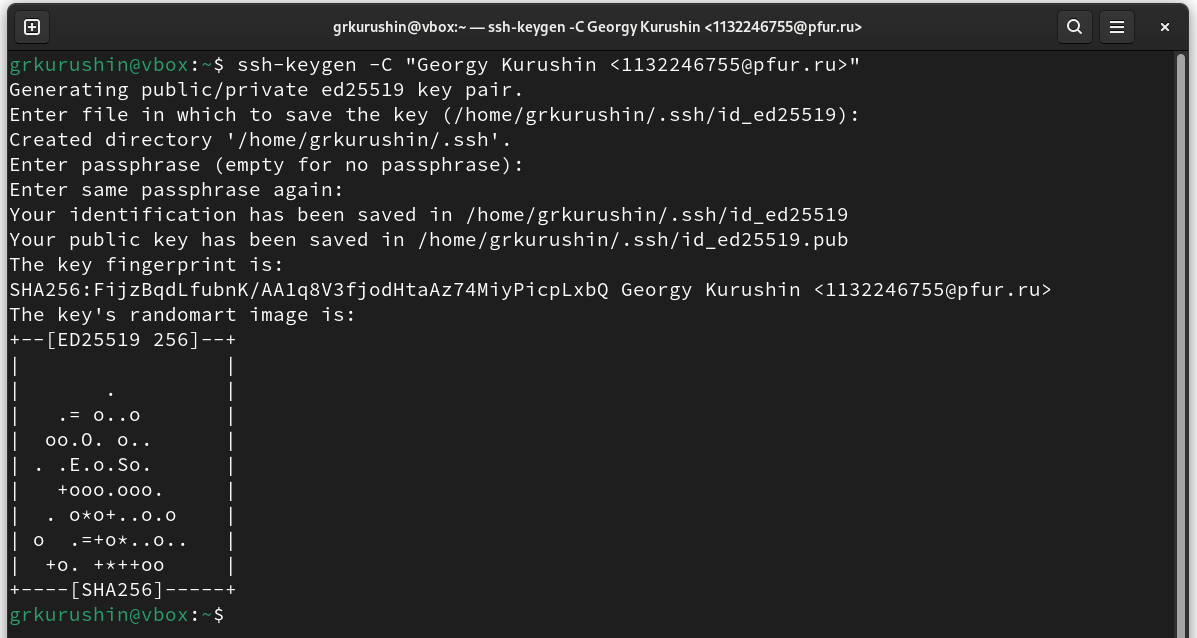
А также ввожу autocrlf и safecrlf (рис. 4.2.4)



*рис.4.2.4. Ввод команд autocrlf и safecrlt.*

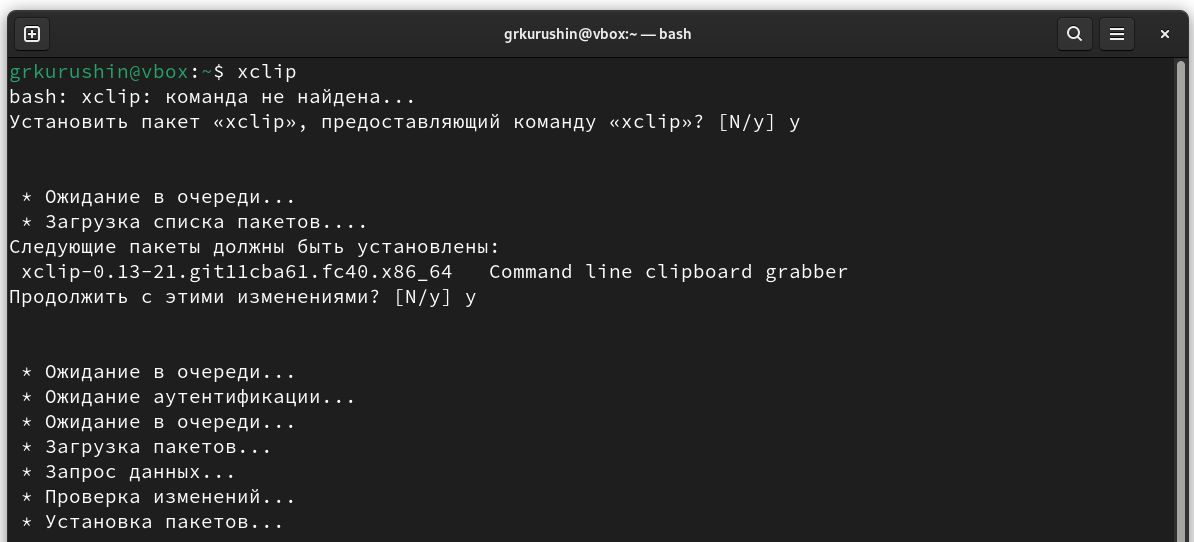
## Создание SSH ключа

Для последующей идентификации пользователя на сервере репозиториев генерирую пару ключей (приватный и открытый). (рис. 4.3.1)



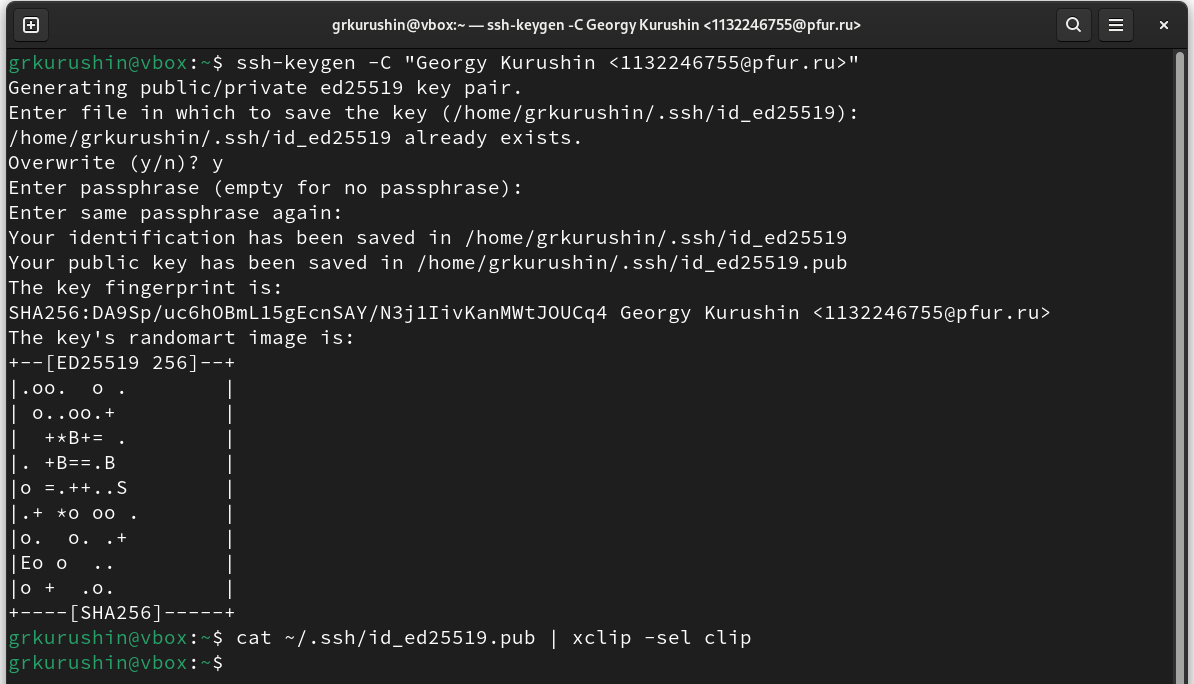
*рис.4.3.1. Генерация ключей.*

Чтобы скопировать из локальной консоли ключ в буфер обмена, устанавливаю команду xclip (рис. 4.3.2).



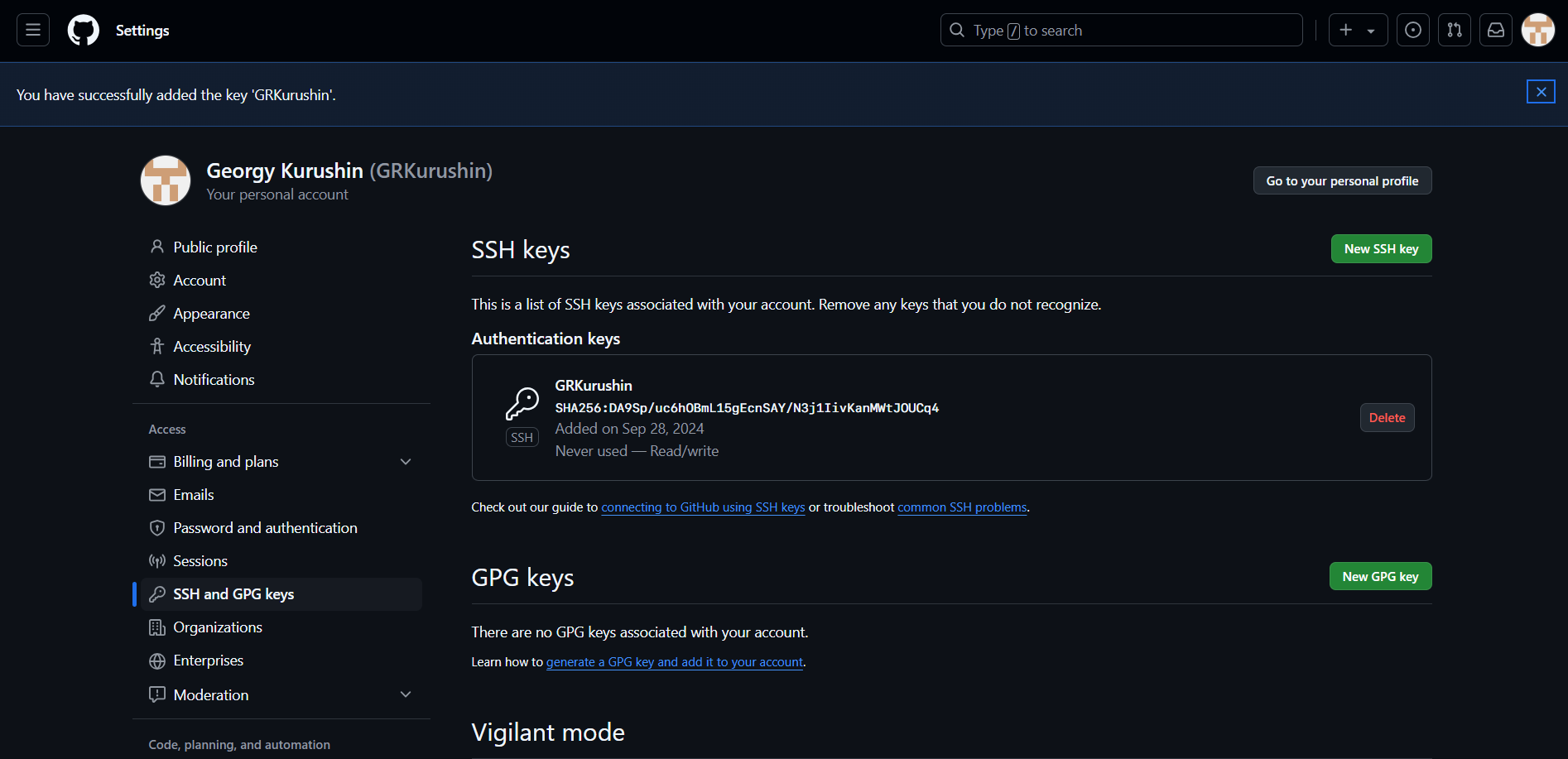
*рис.4.3.2. Скачивание команды xclip.*

Теперь воспользуюсь командой xclip (рис. 4.3.3).



*рис.4.3.3. Копирование ключа.*

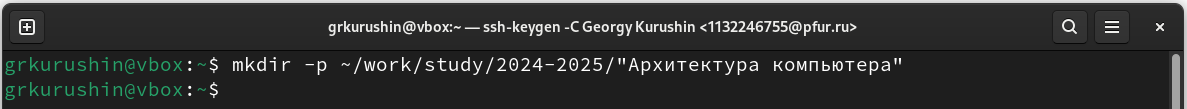
Вставляю ключ в появившееся на сайте поле, указываю его имя. (рис. 4.3.4)



*рис.4.3.4. Вставка ключа.*

## Создание рабочего пространства, репозитория курса на основе шаблона.

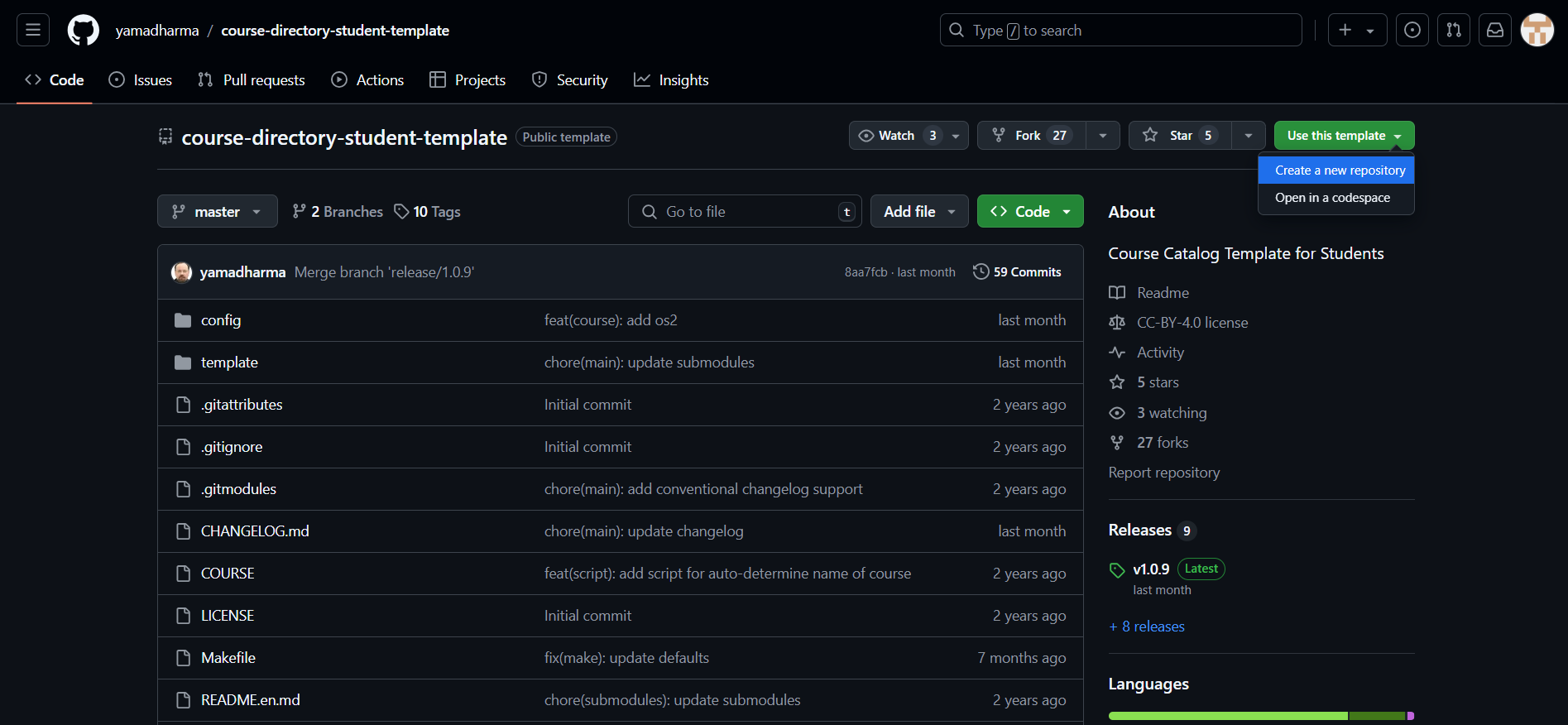
Открываю терминал и создаю репозиторий для предмета «Архитектура компьютера». (рис. 4.4.1.)



*рис.4.4.1. Создание терминала для предмета «Архитектура компьютера».*

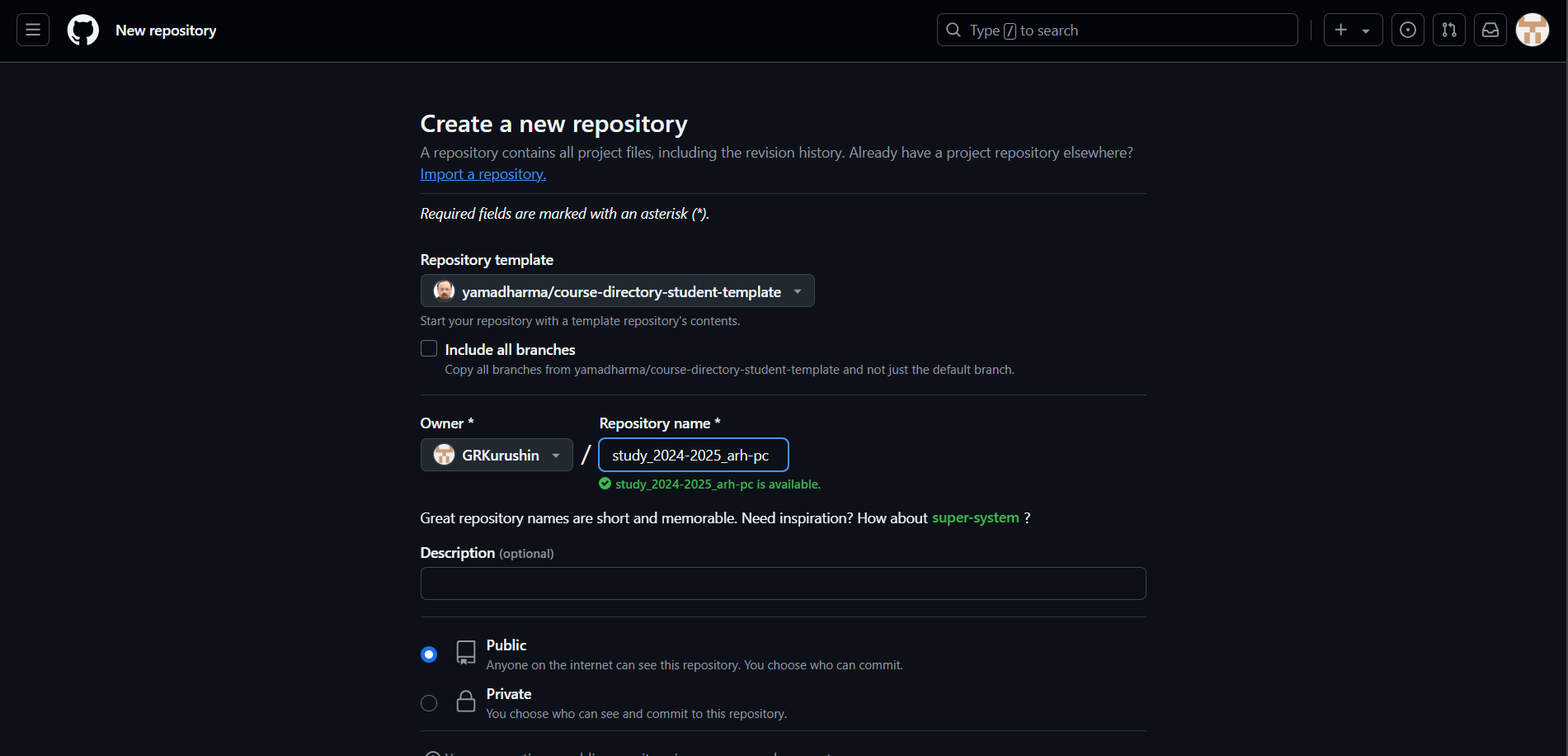
## Создание репозитория курса на основе шаблона.

Захожу на страницу репозитория с шаблоном курса, выбираю его в качестве своего нового. (рис. 4.5.1.)



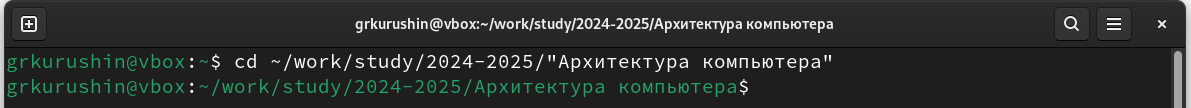
*рис.4.5.1. Выбор шаблона*

Далее создаю его, задав ему имя. (рис. 4.5.2.)



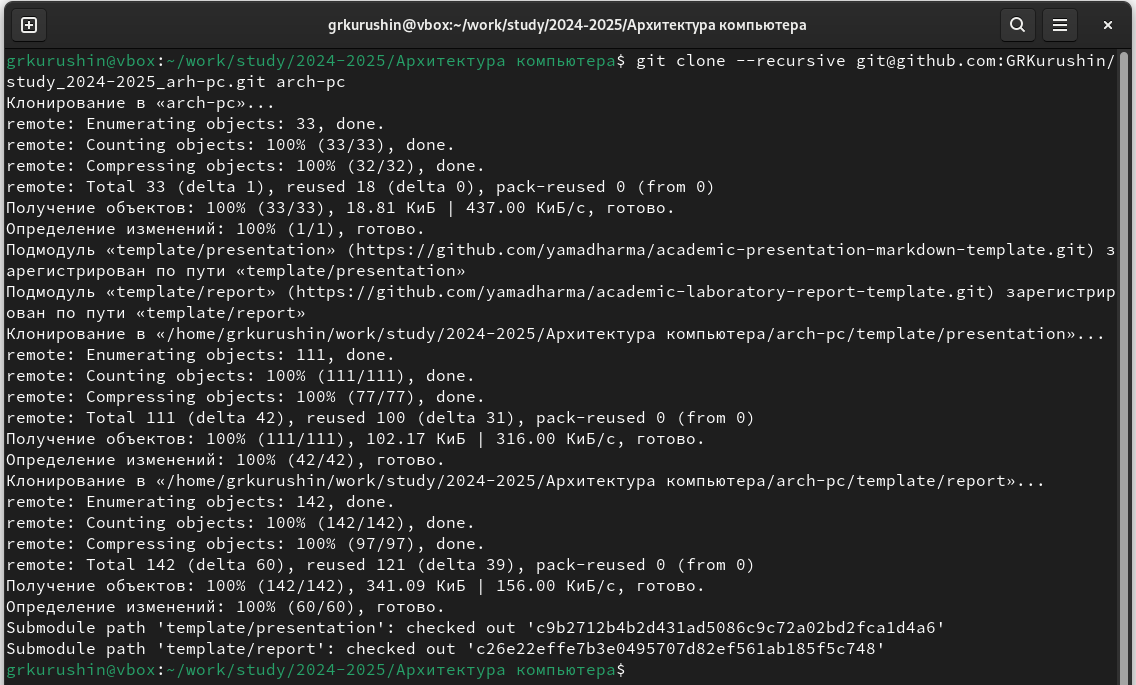
*рис.4.5.2. Создание репозитория.*

Открываю терминал и перехожу в каталог курса. (рис. 4.5.3).



*рис.4.5.2. Переход в каталог курса.*

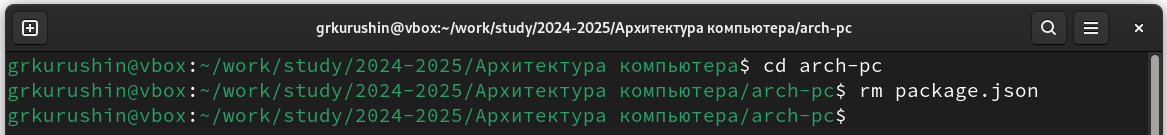
Клонирую созданный репозиторий (рис. 4.5.4.)



*рис.4.5.4. Клонирование репозитория.*

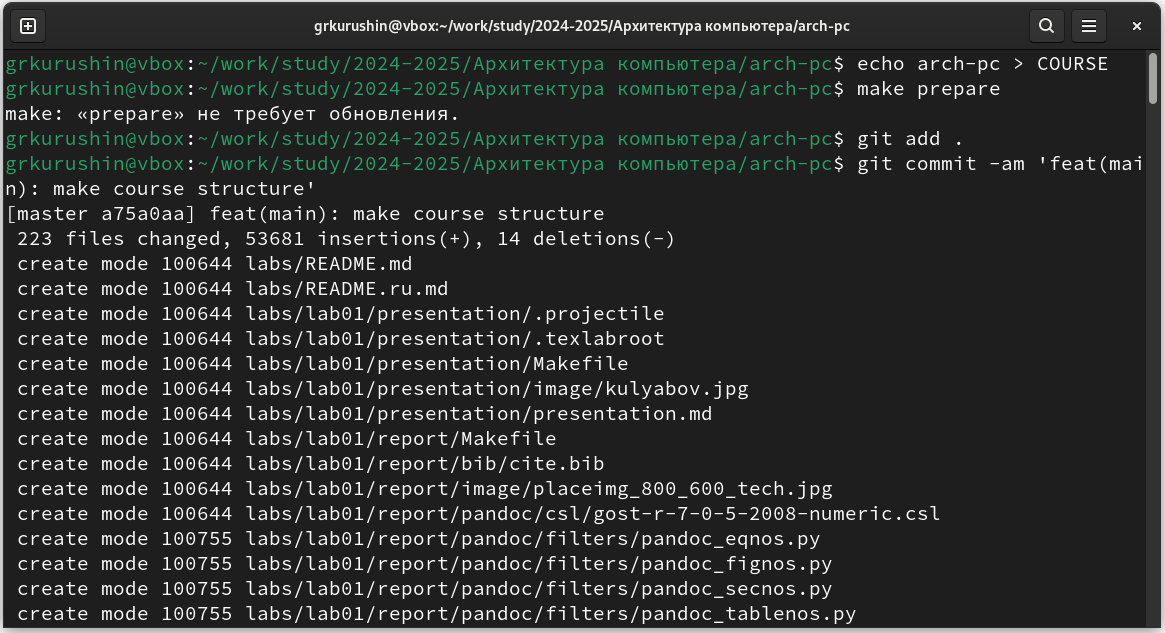
## Настройка каталога курса.

Перехожу в каталог курса и удаляю лишние файлы. (рис. 4.5.5.)



*рис.4.5.5. Удаление лишних файлов в каталоге курса.*

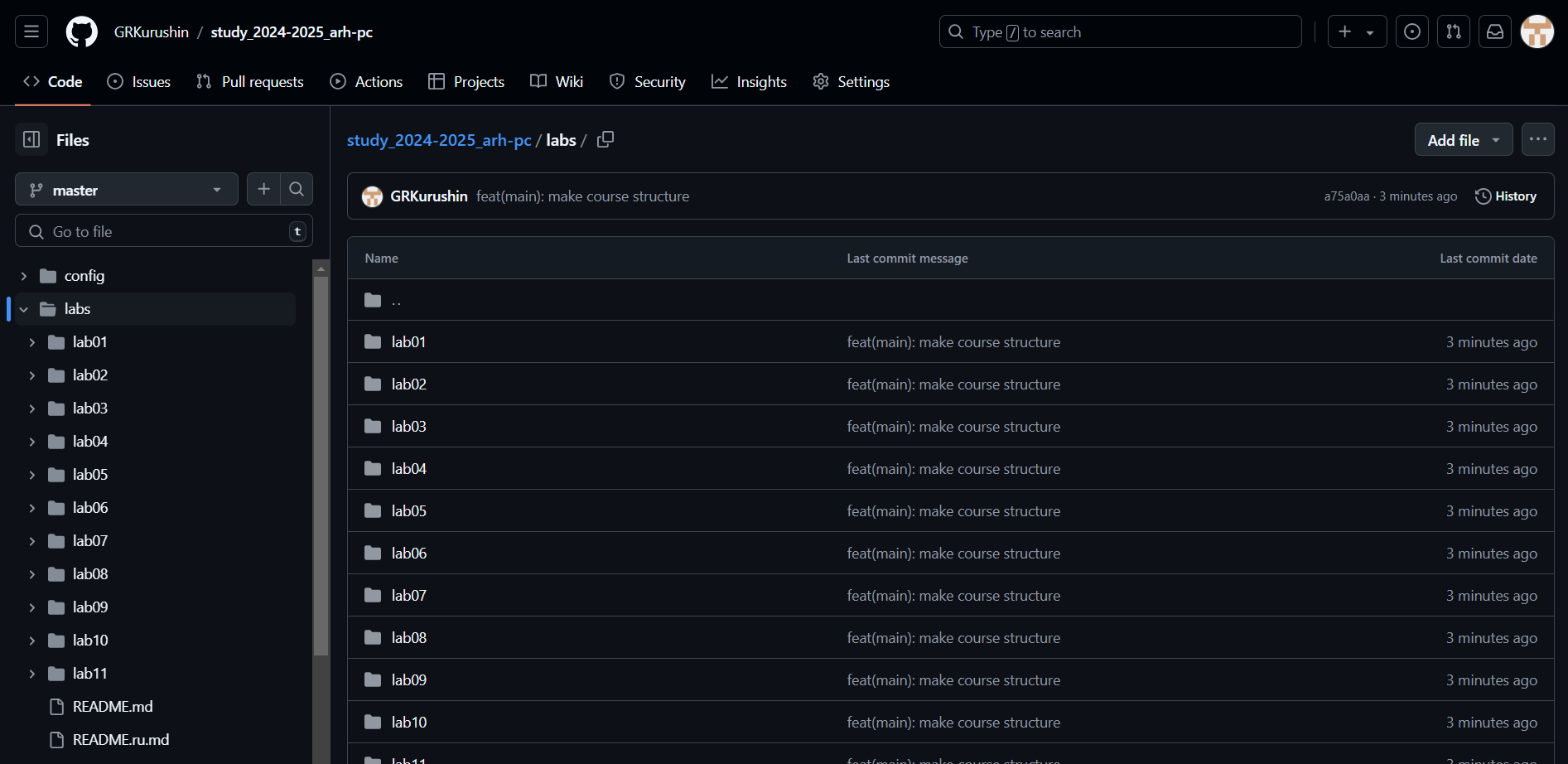
Создаю необходимые каталоги, отправляю файлы на сервер. (рис. 4.5.6.)



*рис.4.5.6. Создание каталогов и их отправка на сервер.*

В локальном репозитории проверяю результат выполненной работы (рис.

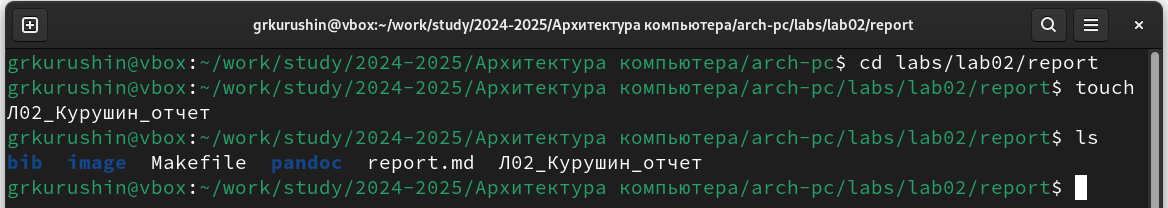
4.5.7)



*рис.4.5.7. Проверка выполненной работы.*

## Выполнение заданий для самостоятельной работы.

Создаю отчет по выполнению второй лабораторной работы в соответствующем каталоге. (рис. 4.7.1) С помощью команды ls проверяю, создан ли файл.



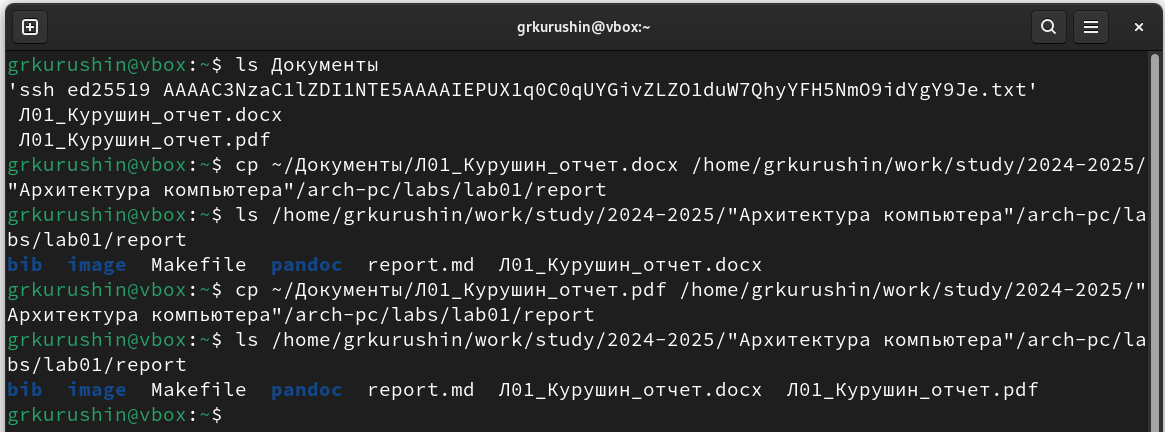
*рис.4.7.1. Создание отчета о выполнении работы.*

Для выполнения второго задания проверяю местонахождение своих лабораторных работ. (рис. 4.7.2)



*рис.4.7.2. Местонахождение лабораторных работ.*

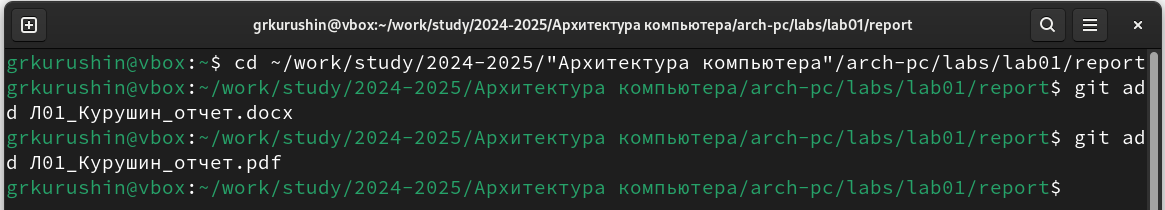
Копирую лабораторную работу с помощью утилиты cp, проверяю местонахождение файлов с помощью команды ls. (рис.4.7.3)



*рис.4.7.3. Копирование отчета по лабораторной работе в нужный каталог.*

Для того чтобы загрузить эти файлы на GitHub, в первую очередь я использую

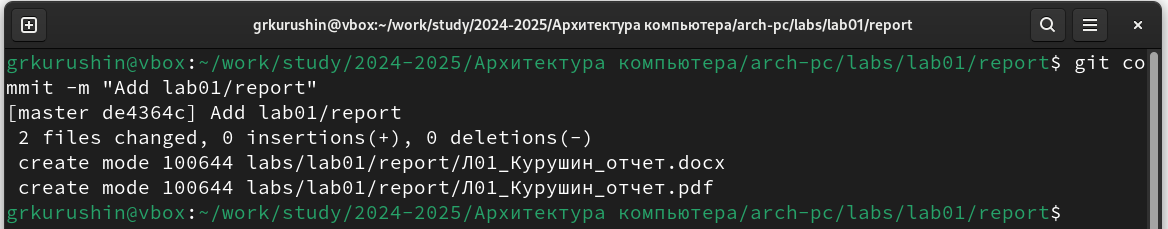
команду git add. Так добавленные мной файлы станут отслеживаемыми. (рис.4.7.4)



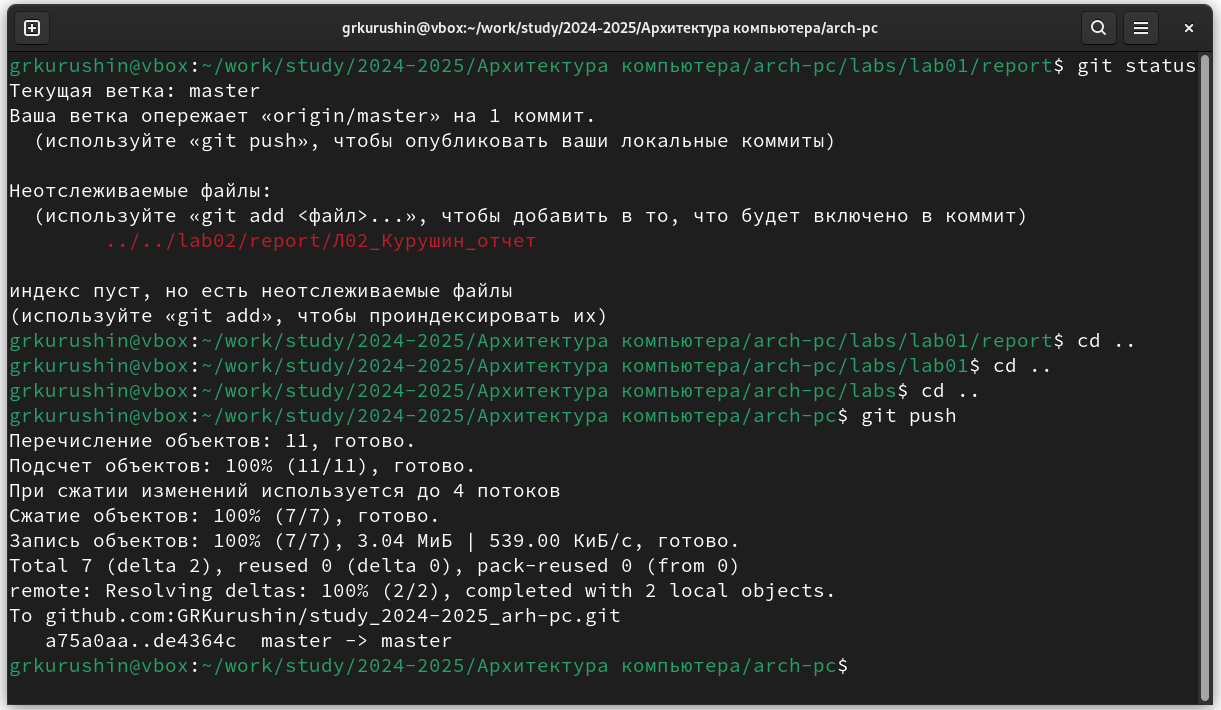
*рис.4.7.4. Добавление файлов с помощью команды git add*

Теперь осуществляю полноценный перенос файлов с помощью команды git

commit -m “…” (рис. 4.7.5.)



*рис.4.7.5. Поручаю консоли совершить изменения.*

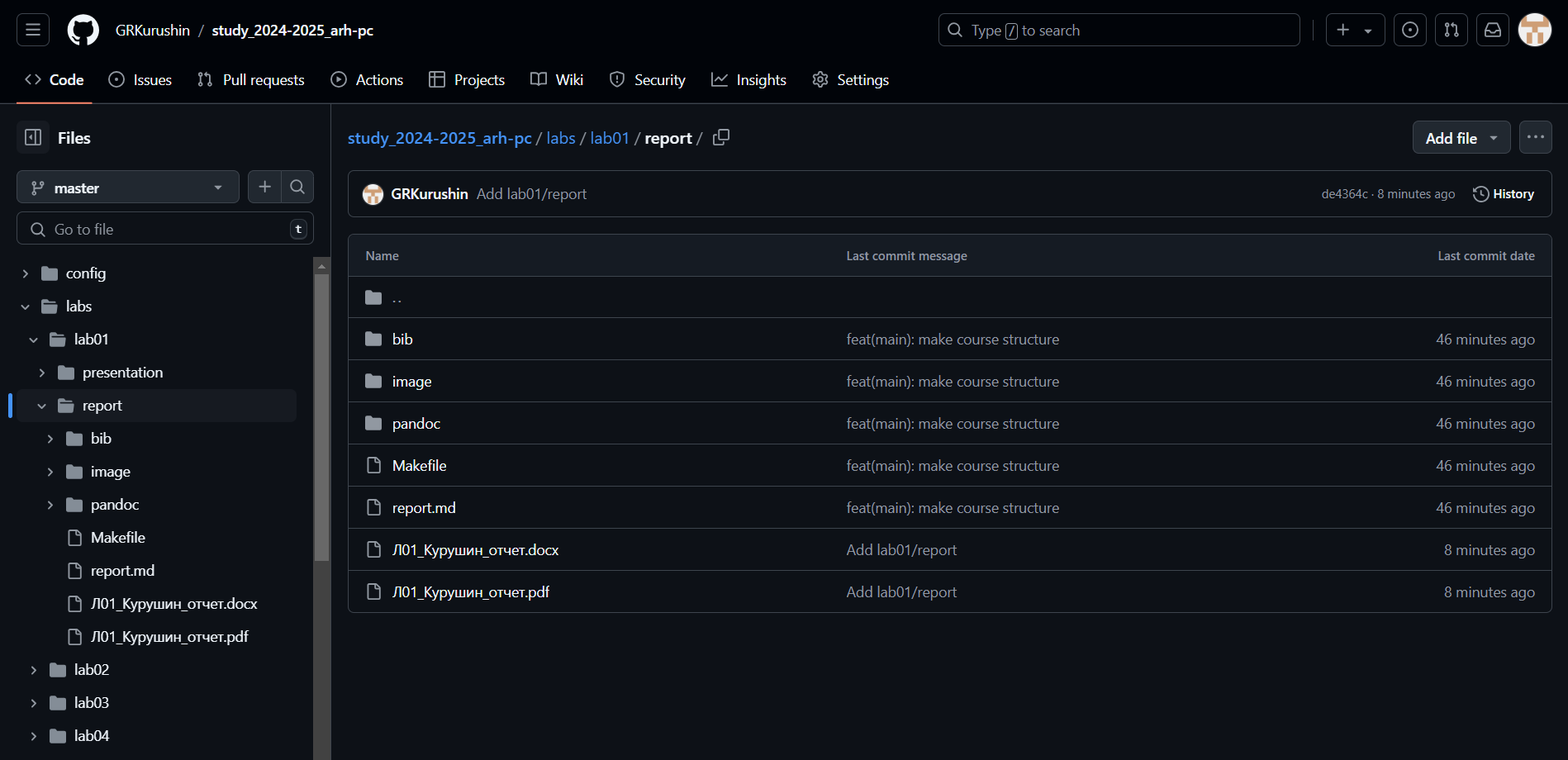


Использую команды: git status и git push, чтобы опубликовать свои локальные

коммиты. (рис. 4.7.6.)

*рис.4.7.6. Команды git status и git push для завершения копирования*

Перехожу в каталоги на GitHub, чтобы убедиться в том, что файлы находятся в нужных репозиториях. (рис. 4.7.7)



*рис.4.7.7. Проверка проделанных операций.*

# Заключение

В заключение хочется отметить, что данная лабораторная работа позволила мне научиться работать с системой Git. Я практиковал свои навыки в работе с командной строкой, теперь уже связывая выполнимое с директориями GitHub.

# Список используемой литературы

1. Архитектура ЭВМ

[https://esystem.rudn.ru/pluginfile.php/2089082/mod\_resource/content/0/Лаборато](https://esystem.rudn.ru/pluginfile.php/2089082/mod_resource/content/0/Р›Р°Р±РѕСЂР°С‚РѕСЂРЅР°СЏ%20СЂР°Р±РѕС‚Р°%20в) [рная%20работа%20№2.%20Система%20контроля%20версий%20Git.pdf](https://esystem.rudn.ru/pluginfile.php/2089082/mod_resource/content/0/Р›Р°Р±РѕСЂР°С‚РѕСЂРЅР°СЏ%20СЂР°Р±РѕС‚Р°%20в)

1. 30 команд Git, необходимых для освоения интерфейса командной строки Git / Хабр <https://habr.com/ru/companies/ruvds/articles/599929/>
2. Система контроля версий: определение, функции, популярные решения <https://gb.ru/blog/sistema-kontrolya-versij/>