# Лабораторная работа № 2:

**Система контроля**

**версий Git**

**ФИО**: Акрур Имад

**Студенческий Билет :** 1032239342

# **1.1 Цель лабораторной работы:**

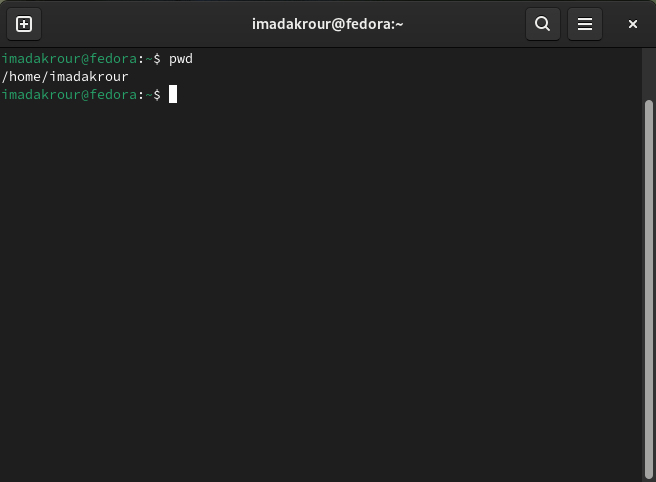
Целью работы является изучить идеологию и применение средств контроля версий. При

обрести практические навыки по работе с системой **git**.

# 1.3 выполнения заданий **лабораторной работы** и результаты:

## 1.4.1. Создание учетной записи на GitHub:

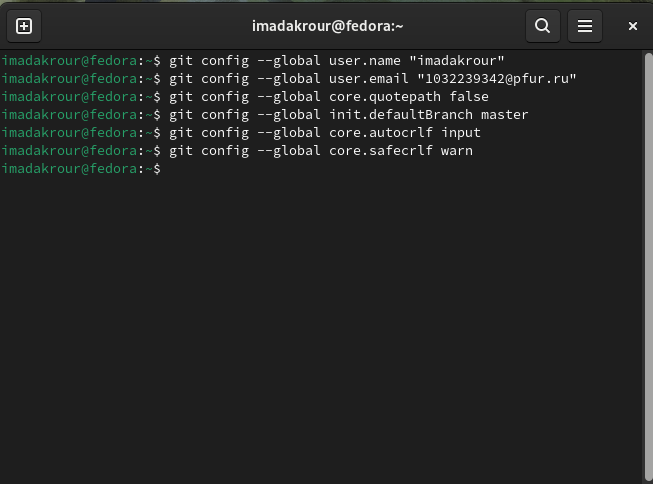
### **Описание задания:** Для начала необходимо создать учетную запись на сайте **GitHub**. Это позволит получить доступ к управлению репозиториями для работы над проектами.



*в моем случае учетная запись уже существует, поэтому этот шаг был пропущен.*

## 1.4.2. Базовая настройка Git**:**

1. **Указание имени пользователя и email для создания коммитов в репозиториях. Эти данные будут добавляться к каждому коммиту, чтобы идентифицировать автора изменений.:**

****

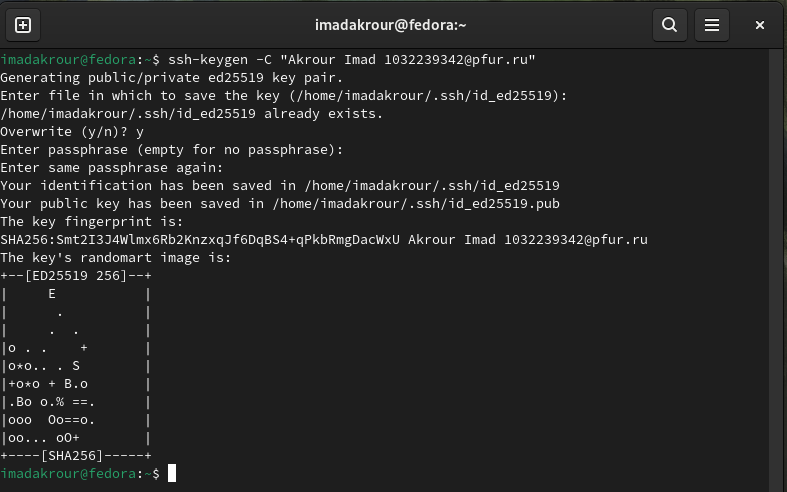
*Эта последовательность команд настраивает имя пользователя, email, кодировку UTF-8, задает начальную ветку master, устанавливает обработку концов строк (autocrlf), и включает предупреждения о несоответствиях концов строк (safecrlf). Выполнение всех команд в одной строке позволяет быстрее закончить настройку и сразу перейти к работе с Git. Все параметры будут применены глобально для всех будущих репозиториев.*

## 1.4.3. Создание SSH-ключа:

**Описание выполняемого задания:**

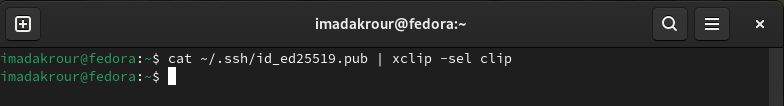
Для безопасного подключения к репозиториям на GitHub, нужно сгенерировать SSH-ключи (публичный и приватный). Это позволит вам работать с репозиториями, не вводя каждый раз логин и пароль.

1. **Генерация SSH-ключа:**

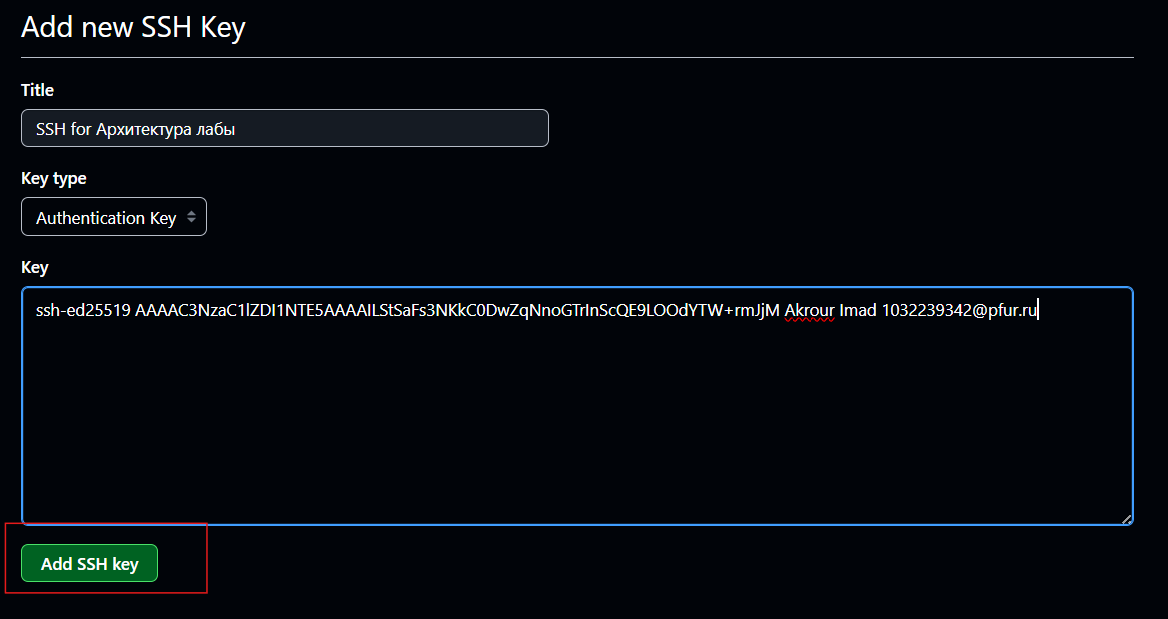
******

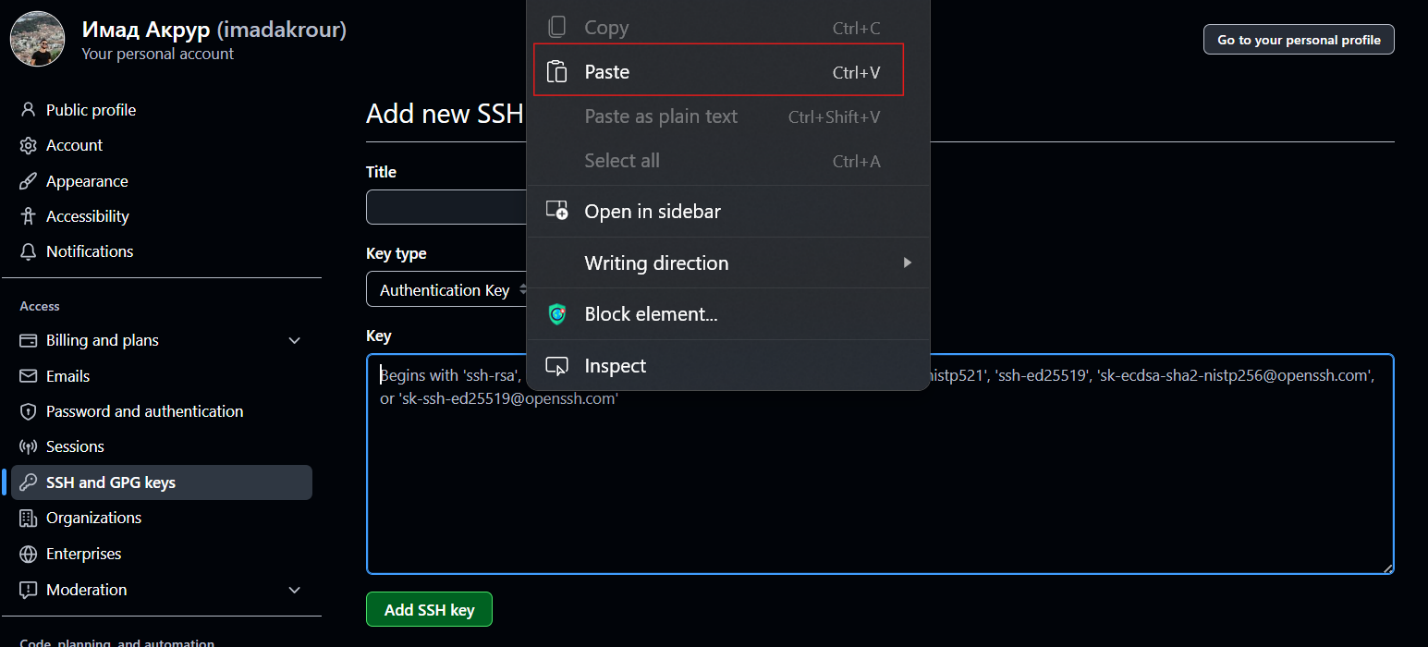
Каталог ~/.ssh/ — это стандартное место для хранения SSH-ключей. Не изменяйте путь, если вы не хотите использовать другое место для хранения.

**Копирование публичного ключа в буфер обмена:**

****

*Команда cat ~/.ssh/id\_rsa.pub выводит содержимое публичного ключа, а команда xclip -sel clip копирует это содержимое в буфер обмена*

****

****

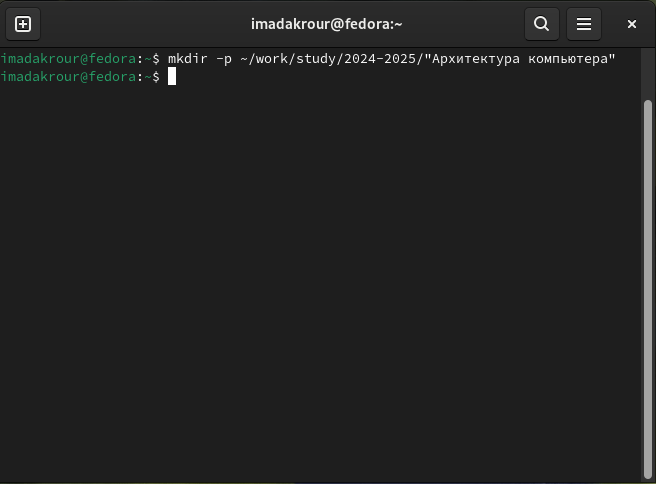
*вставить ключ на сайт GitHub.*

## 1.4.4. Создание рабочего пространства и репозитория курса на основе шаблона:

Для правильной организации рабочих файлов и проектов в рамках курса необходимо создать рабочее пространство по определенной структуре. В этом шаге будет выполнено создание директории для предмета «Архитектура компьютера», а также будет продемонстрировано, как структурировать папки для лабораторных работ.

**Описание выполняемого задания:**

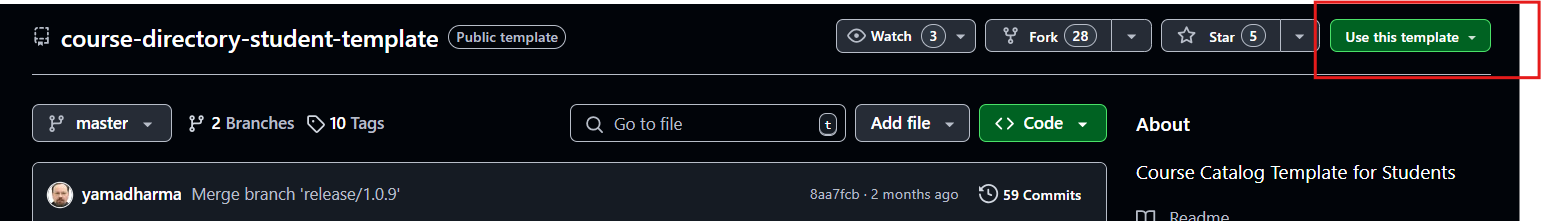
**Создание структуры рабочего пространства :**

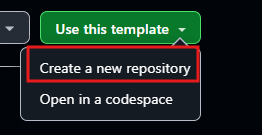
****

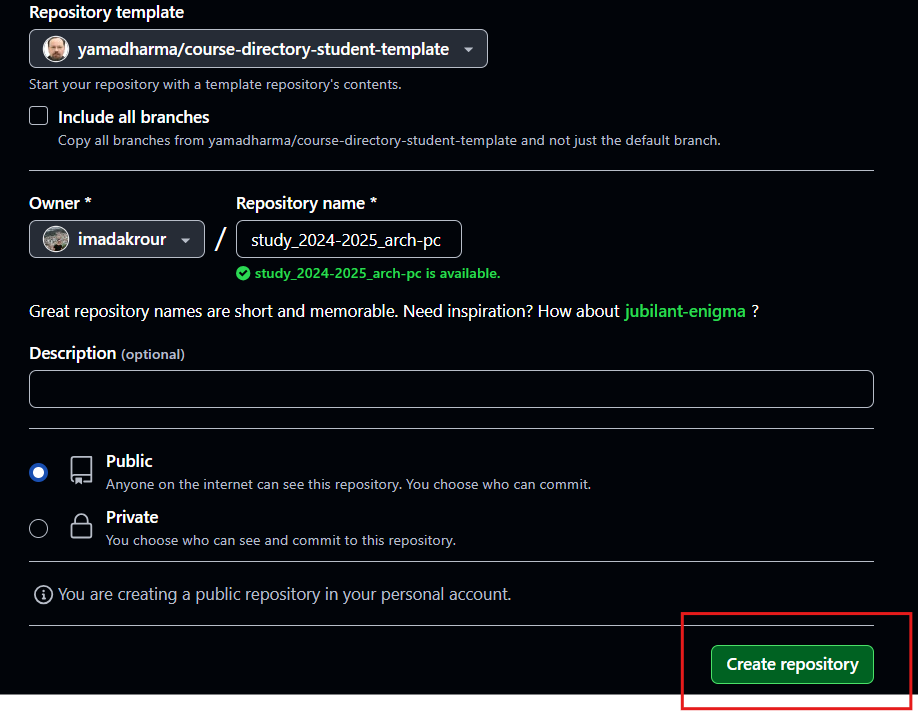
*Эта команда создаст нужные каталоги по иерархии. Опция -p создает промежуточные каталоги, если они еще не существуют*

## 1.4.5. Сознание репозитория курса на основе шаблона:



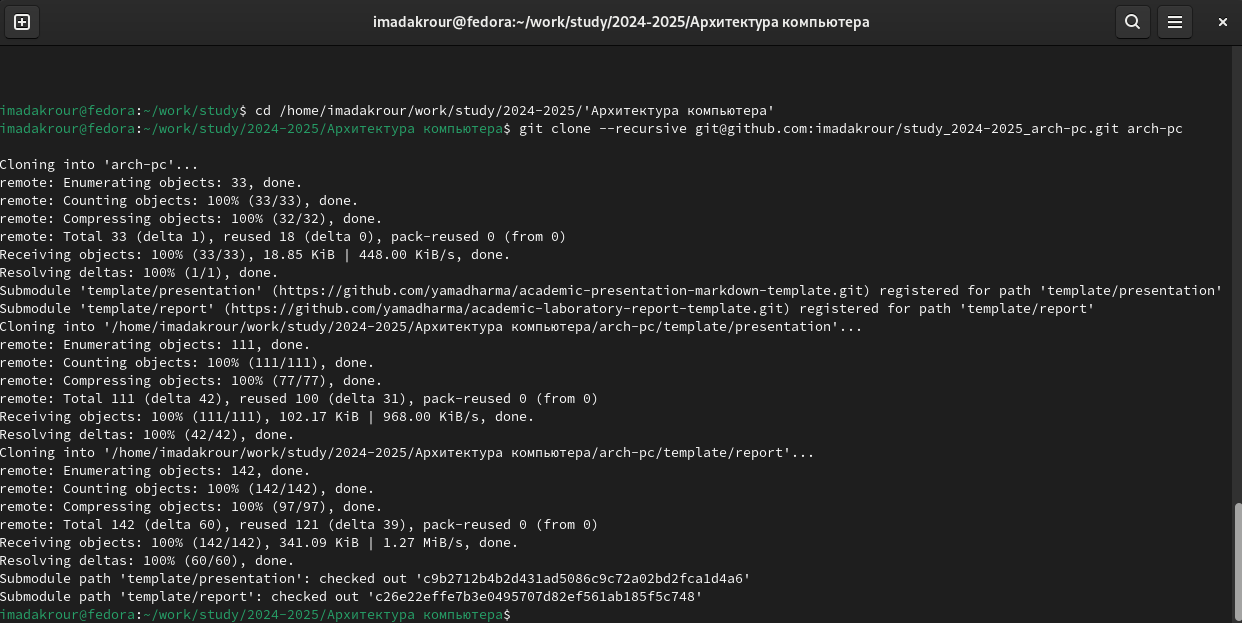




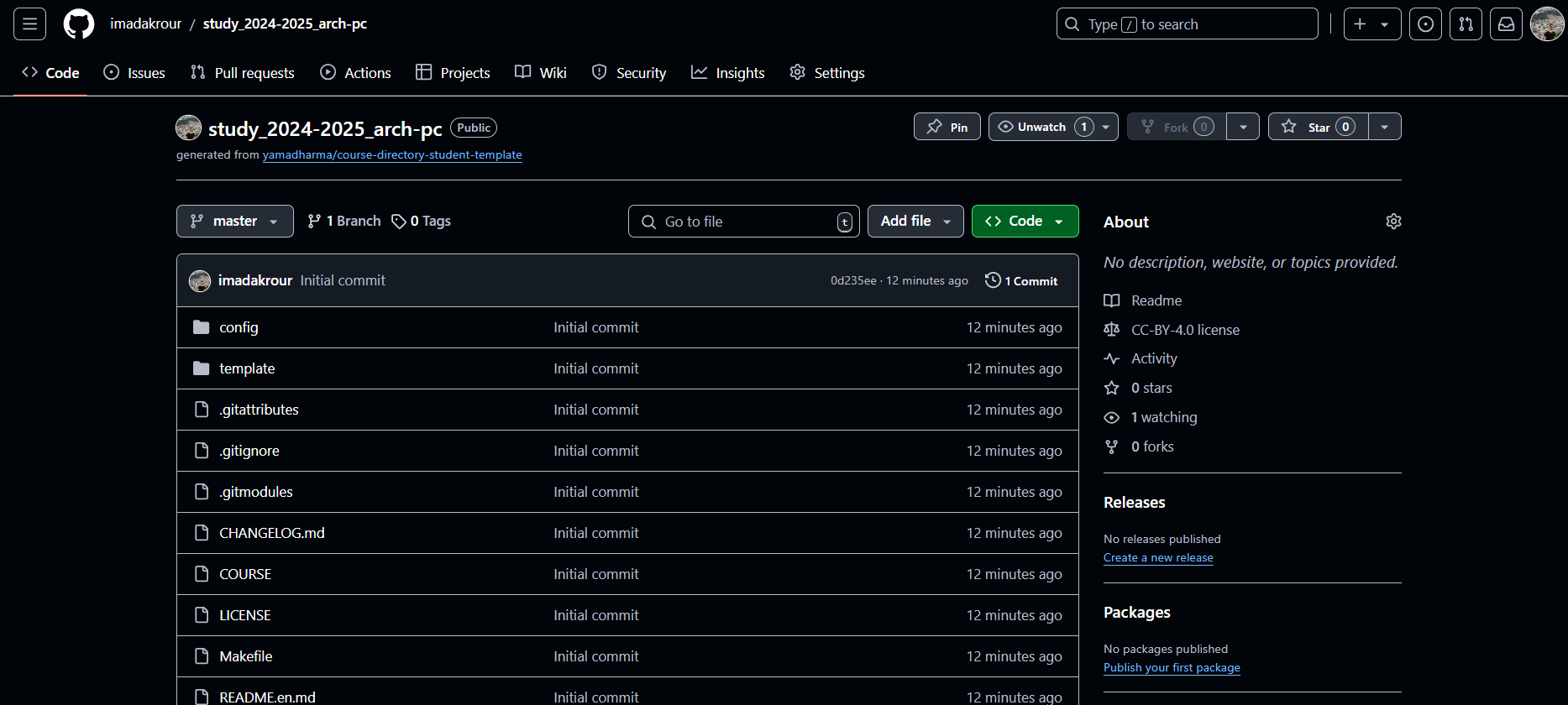


*Эти действия создают новый репозиторий на основе предоставленного шаблона, который содержит структуру и необходимые файлы для работы по курсу.*

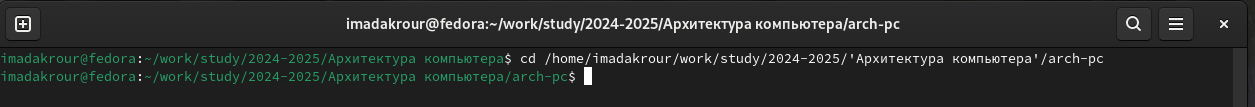
**Клонирование репозитория на локальный компьютер :**

****

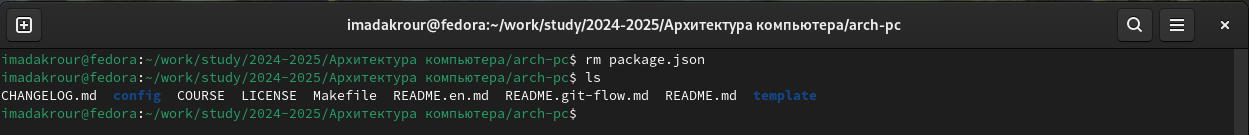
*Команда git clone --recursive позволяет загрузить все файлы из удаленного репозитория в папку arch-pc.*

****

**Настройка каталога курса :**

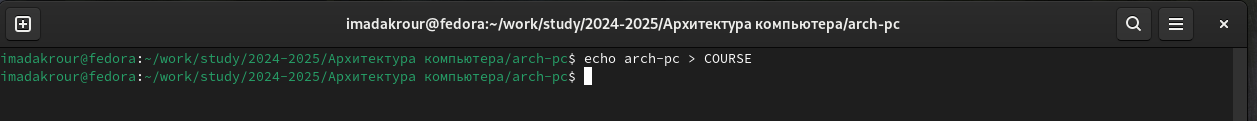
****

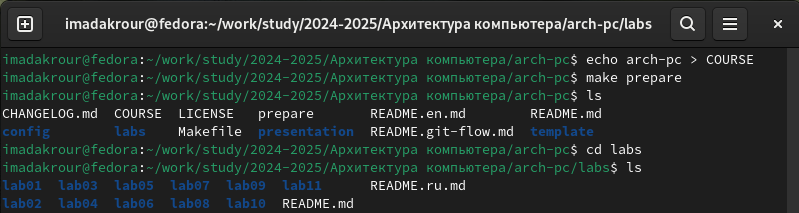
**Удаление ненужного файла package.json** **:**

****

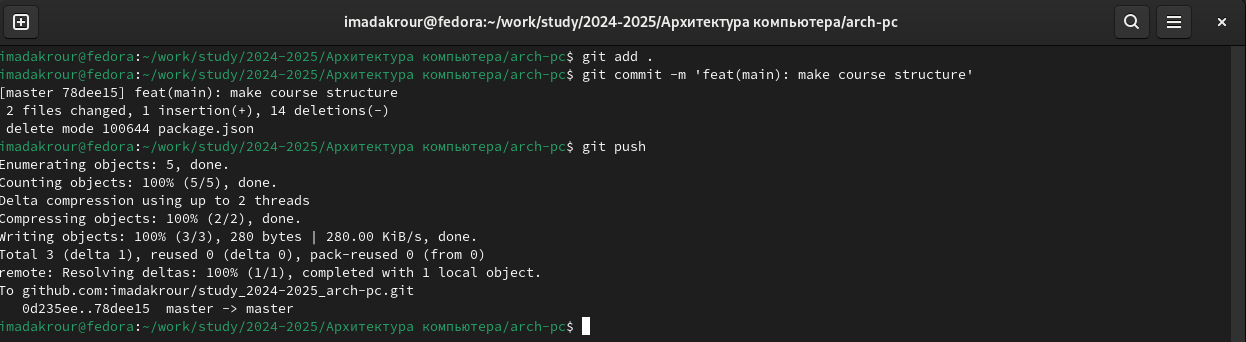
*Удаление лишних файлов помогает избежать путаницы и оставить только необходимые для курса файлы.*

**Создание файла COURSE с названием курса:**

****

******

**Отправка изменений на сервер :**

****

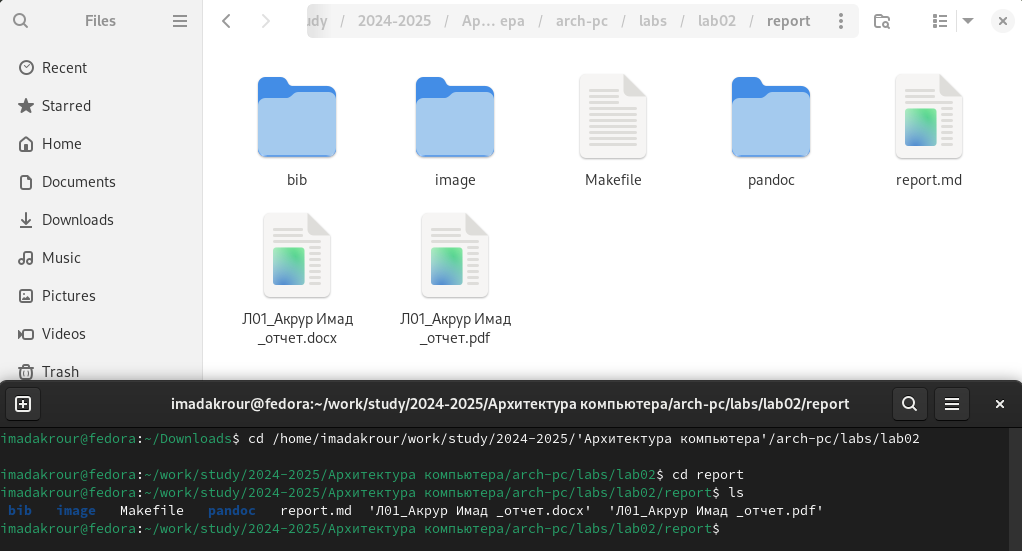
*Эти команды добавляют изменения в локальный репозиторий, создают коммит с описанием и отправляют изменения в удаленный репозиторий на GitHub.*

# 1.5 Описание результатов выполнения заданий для **самостоятельной работы**:

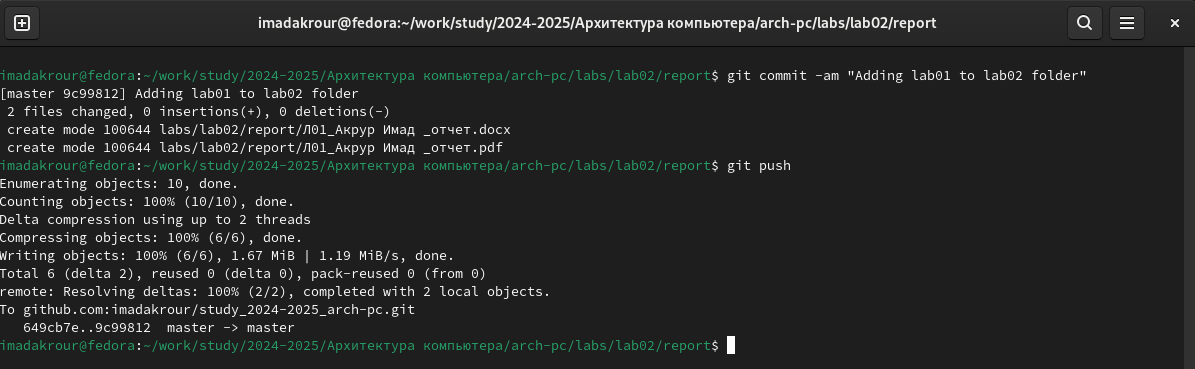
## 1. Создание отчета в соответствующем каталоге рабочего пространства:

**Описание задания:**

Данное задание включает в себя создание отчета о выполнении лабораторной работы, копирование предыдущих отчетов и загрузку файлов на GitHub.



*Я скопировал файлы предыдущего отчета в папку lab02, используя команду cp. Сначала я перешел в каталог с файлами отчета с помощью команды cd, а затем выполнил команду для копирования всех файлов*



*После копирования файлов я выполнил команды для отправки их на GitHub*

**

*Затем я сделал скриншот в GitHub, чтобы показать, что файлы были обновлены и теперь отображаются в репозитории.*

# 1.7 Вопросы для самопроверки:

**1. Что такое системы контроля версий (VCS) и для чего они предназначаются?**

Системы контроля версий (VCS) — это инструменты для управления изменениями в коде и файлах проекта, позволяющие отслеживать историю изменений, работать с разными версиями и сотрудничать в команде.

**2. Объясните понятия VCS: хранилище, commit, история, рабочая копия**

- Хранилище (repositoryместо, где хранятся все версии проекта.

- Commitзафиксированные изменения с комментарием, представляющие версию проекта.

- История после довательность всех commit

- Рабочая копия (working copyлокальная версия проекта, с которой работает разработчик.

**3. Чем отличаются централизованные и децентрализованные VCS? Примеры.**

- Централизованные VCS (например, SVN): одно центральное хранилище, доступное для всех.

- Децентрализованные VCS (например, Git): каждый пользователь имеет своё полное хранилище, с возможностью синхронизации.

**4. Действия с VCS при единоличной работе с хранилищем.**

- Инициализация репозитория.

- Добавление файлов и создание commit

- Проверка состояния (git status), просмотр истории (git log).

**5. Порядок работы с общим хранилищем VCS.**

- Клонирование репозитория.

- Создание ветки, внесение изменений.

- Commit изменений.

- Слияние с основной веткой через pull request.

**6. Основные задачи, решаемые Git.**

Управление версиями, создание веток, слияние изменений, разрешение конфликтов, работа с удалёнными репозиториями.

**7. Команды Git: краткая характеристика.**

- `git init`: создание репозитория.

- `git clone`: клонирование удалённого репозитория.

- `git add`: добавление файлов для отслеживания.

- `git commit`: фиксация изменений.

- `git push`: отправка изменений в удалённый репозиторий.

- `git pull`: получение изменений из удалённого репозитория.

**8. Примеры работы с локальными и удалёнными репозиториями.**

- Локальные : инициализация репозитория, создание commit , проверка состояния.

- Удалённые: клонирование, отправка изменений (git push), получение обновлений (git pull).

# Комментарии и выводы по результатам выполнения заданий:

Выполнение задания помогло закрепить навыки работы с системами контроля версий, организацией файлов в рабочем пространстве, а также загрузкой проектов на GitHub. Эти навыки важны для поддержания эффективной командной работы, обеспечения безопасности данных и удобства доступа к прошлым версиям проектов.