**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:

Мустафина Аделя Юрисовна

Группа:

НКАбд-03-24

Студенческий билет №:

1132246719

**МОСКВА**

2024 г.

Содержание

[1. Цель работы 3](#_Toc178432704)

[2. Задание 4](#_Toc178432705)

[3. Теоретическое введение 5](#_Toc178432706)

[4. Выполнение лабораторной работы 6](#_Toc178432707)

[5. Выводы 18](#_Toc178432708)

# Цель работы

Целью работы является изучить идеологию и применение средств контроля версий. Приобрести практические навыки по работе с системой git.

# Задание

1. Настройка github
2. Базовая настройка git
3. Создание SSH ключа
4. Сознание рабочего пространства и репозитория курса на основе шаблона
5. Сознание репозитория курса на основе шаблона
6. Настройка каталога курса
7. Выполнение заданий

# Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.

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.

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.

# Выполнение лабораторной работы

*2.4.1. Настройка github*

Я создала учетную запись на сайте https://github.com/ и заполнила основные данные, указав имя Adelya Mustafina и aymustafina и свою почту.

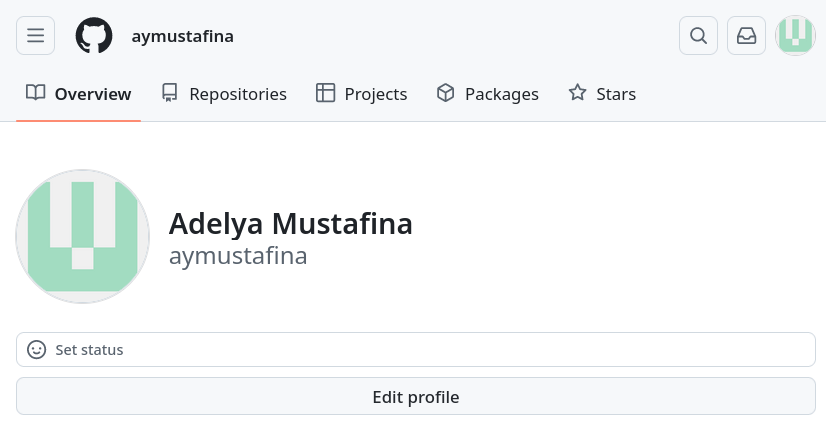


Рис. 4.1.1. Настройка github

*2.4.2. Базовая настройка git*

Делаю предварительную конфигурацию git. Открыв терминал в Fedora, указываю имя и почту, которую вводила для создания аккаунта в Githab.

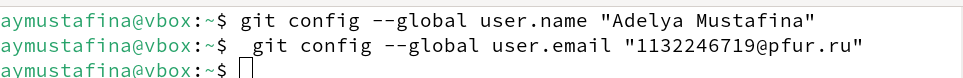


Рис. 4.2.2 Предварительная конфигурация

Настраиваю utf-8.



Рис. 4.2.3 Предварительная конфигурация

Задала имя начальной ветки.



Рис. 4.2.3 Предварительная конфигурация

Параметры autocrlf и safecrlf





Рис. 4.2.4 Предварительная конфигурация

*2.4.3. Создание SSH ключа*

На сервере создам пару ключей для последующей идентификации. Для этого ввожу команду ssh-keygen -C "Имя Фамилия, [work@email](mailto:work@email)".

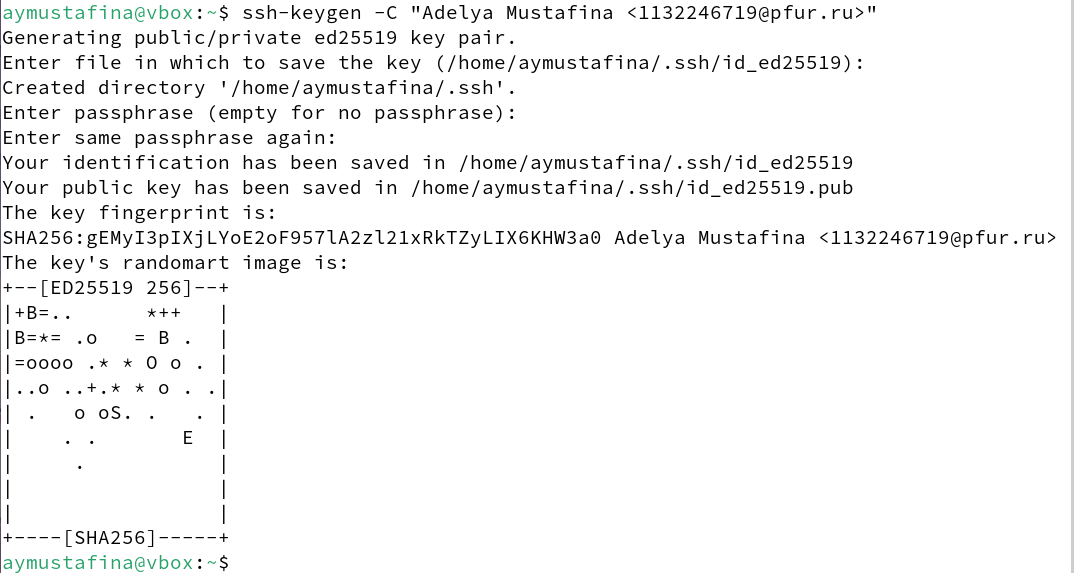


Рис. 4.3.6 Создание ключей

С помощью утилиты xclip скопирую текст через терминал. Но для ее использования необходимо ее еще устанавливаю.

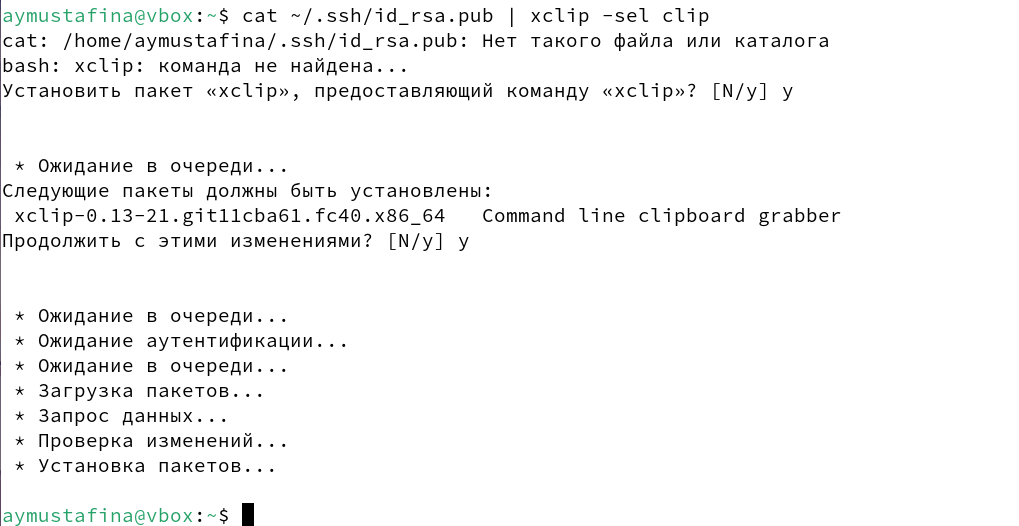


Рис. 4.3.7 Утилита xclip

Копирую текст с помощью утилиты xclip.

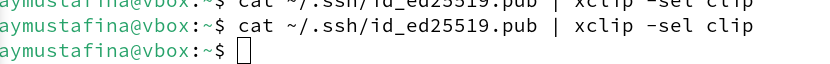


Рис. 4.3.8 Утилита xclip

Захожу на сайт Githab, вхожу в свой аккаунт и открываю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key».

Вставляю свой скопированный ключ в поле «Key». В поле Title указываю имя для ключа. И нажимаю «Add SSH-key», чтобы завершить добавление ключа

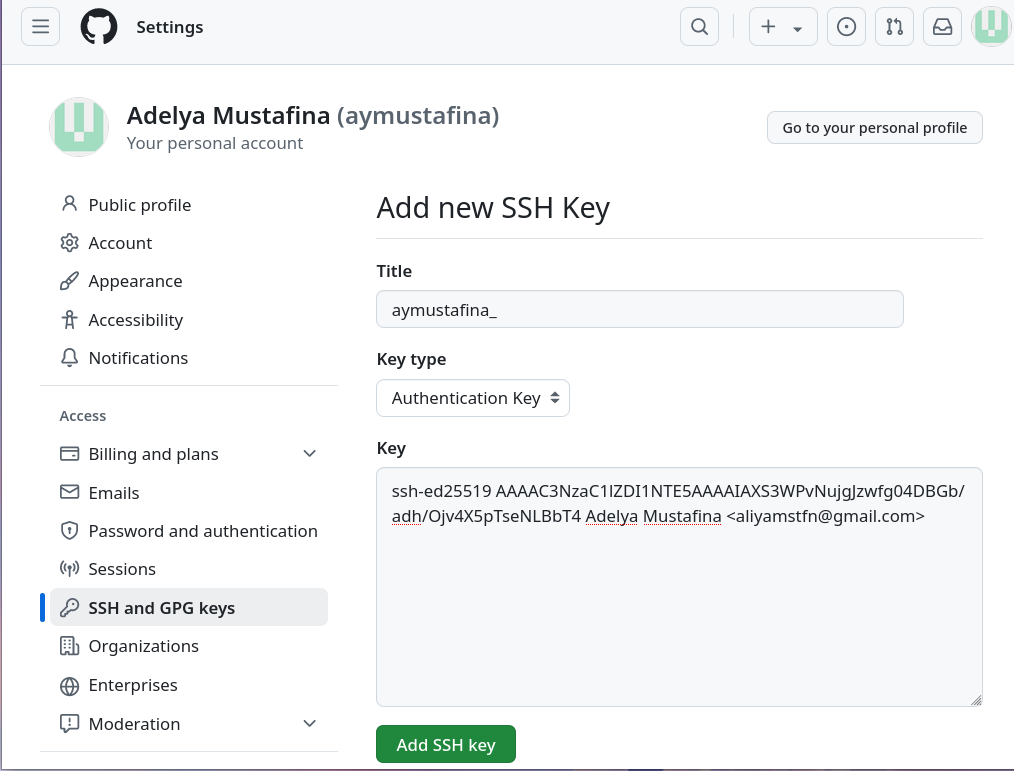


Рис. 4.3.9 Создание ключа

*2.4.4. Создание рабочего пространства и репозитория курса на основе шаблона*

Я создаю новую директорию с помощью команды mkdir и ключа –p.

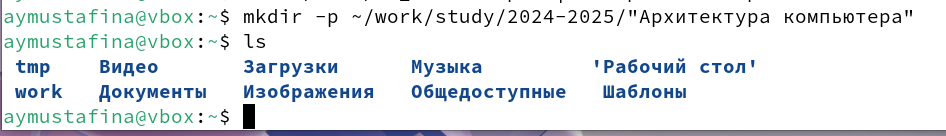
**

Рис. 4.4.10 Создание директории

*2.4.5. Создание репозитория курса на основе шаблона*

Перехожу на страницу репозитория с шаблоном курса https://github.com/yamadharma/cour se-directory-student-template. Нажимая на кнопку Use this template, в открывшемся окне и создаю репозиторий Create repository from template и задаю имя репозитория (Repository name) study\_2023–2024\_arhpc.

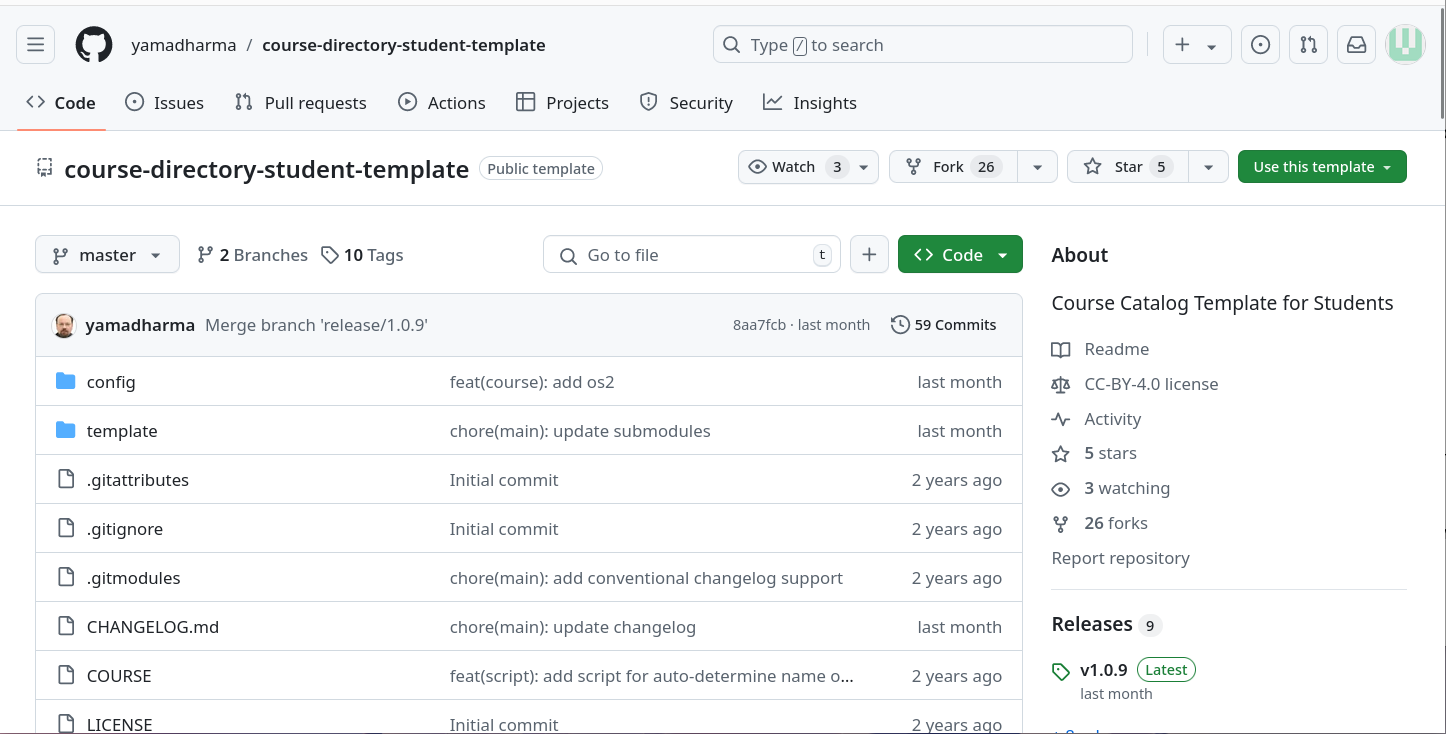


Рис. 4.5.11 Шаблон курса

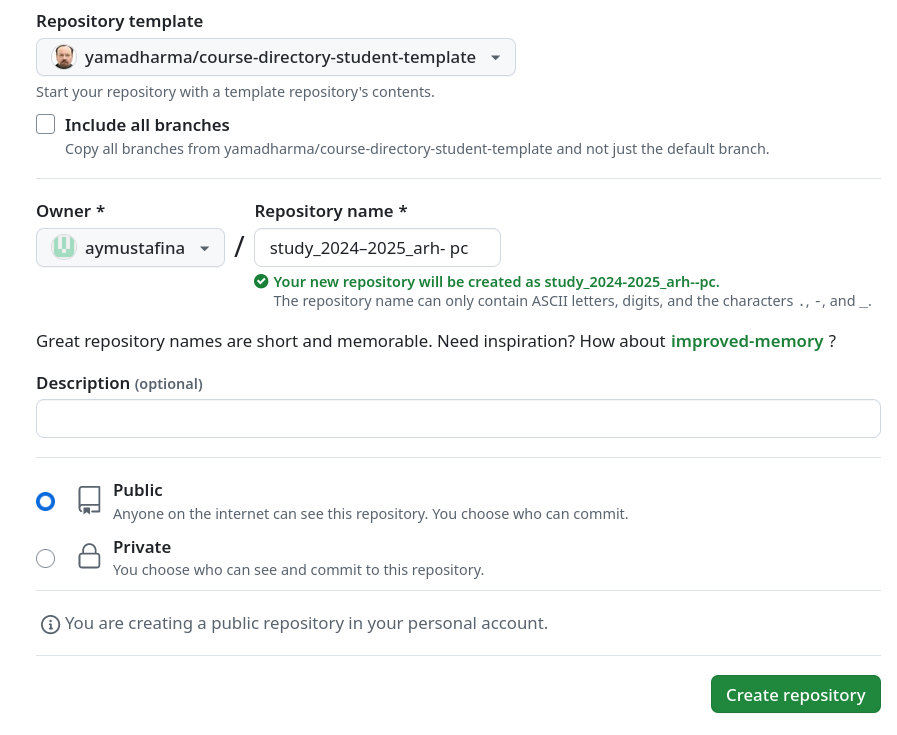


Рис. 4.5.12 Шаблон курса

Репозиторий создан.

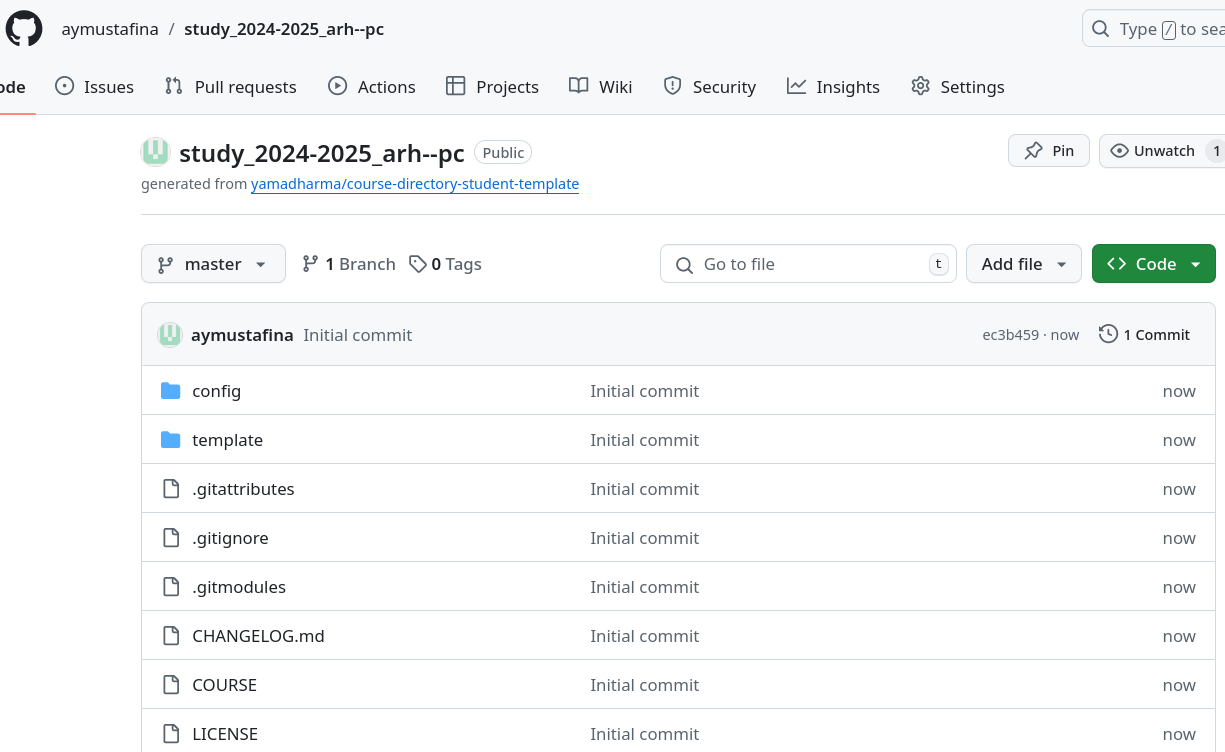


Рис. 4.5.13 Шаблон курса

Через терминал перейдем в созданный каталог.



Рис. 4.5.14 Перемещение в каталог

Копирую созданный репозиторий.

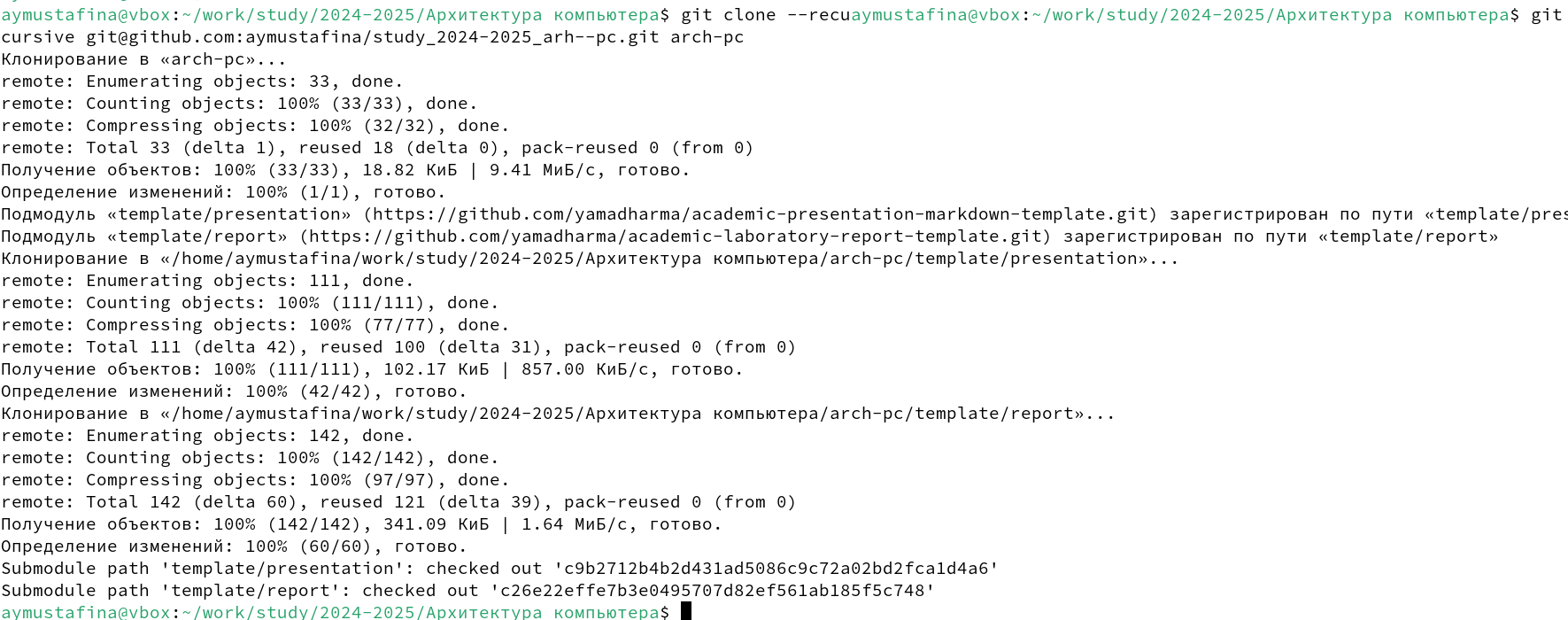


Рис. 4.5.15 Копирование репозитория

Копирую ссылку для клонирования на странице созданного репозитория, в начале перейдя в окно code.

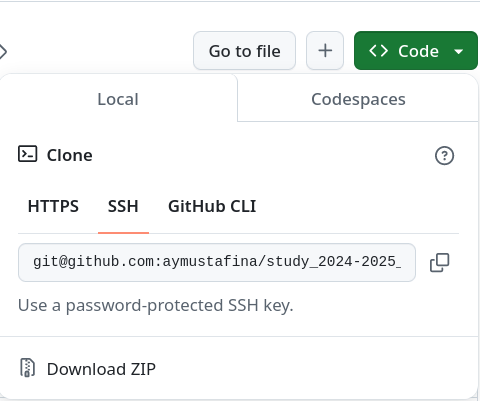


Рис. 4.5.16 Копирование ссылки

*2.4.6. Настройка каталога курса*

Перехожу в каталог arch-pc с помощью команды cd.

Удаляю лишние файлы с помощью команды rm.

Создаю необходимые каталоги. Далее отправляю все созданные каталоги с локального репозитория на сервер.

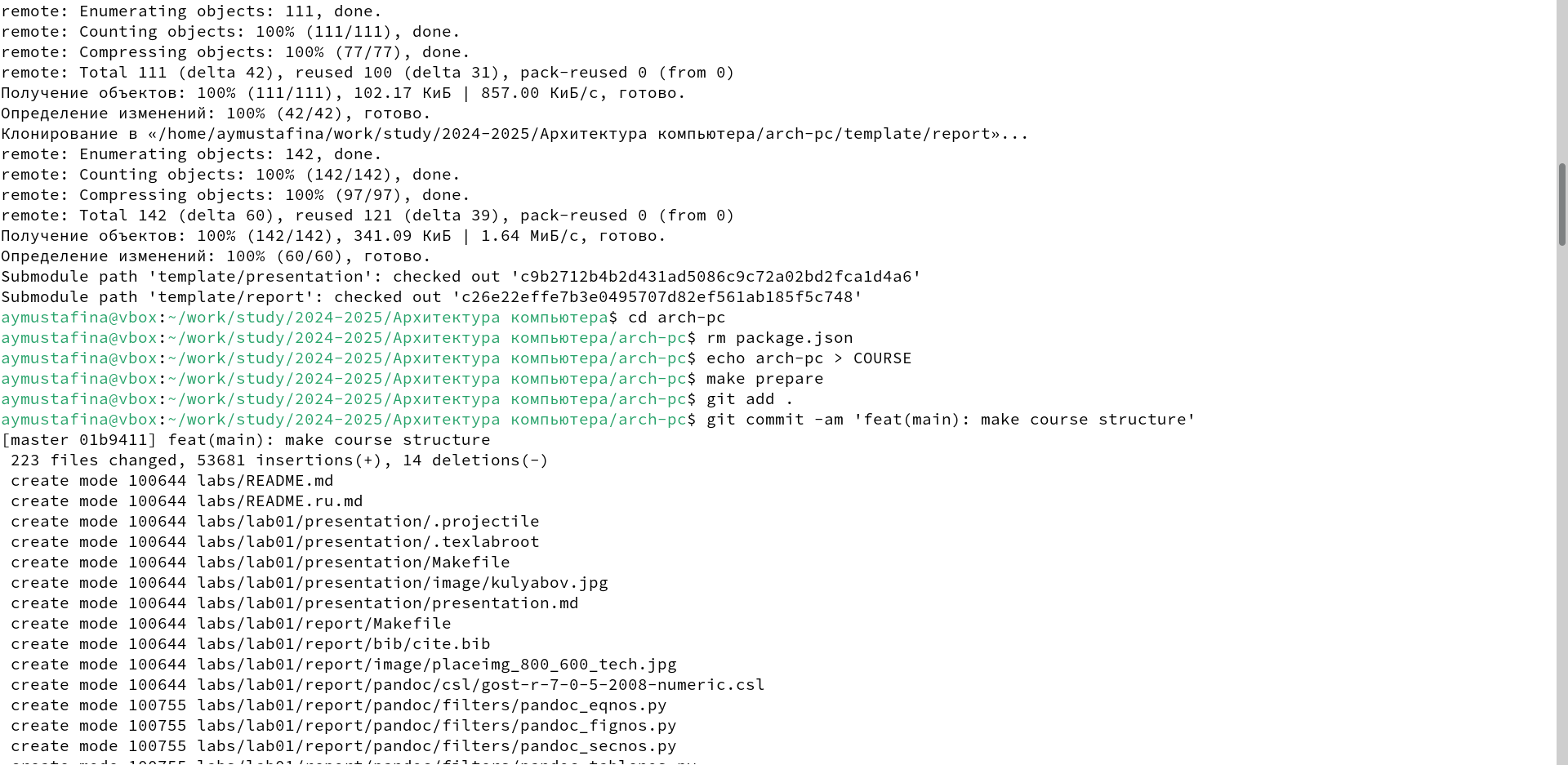
**

Рис. 4.6.17 Изменение курса.

Добавляю созданные каталоги с помощью git add, комментирую и сохраняю изменения на сервере как добавление курса с помощью git commit.

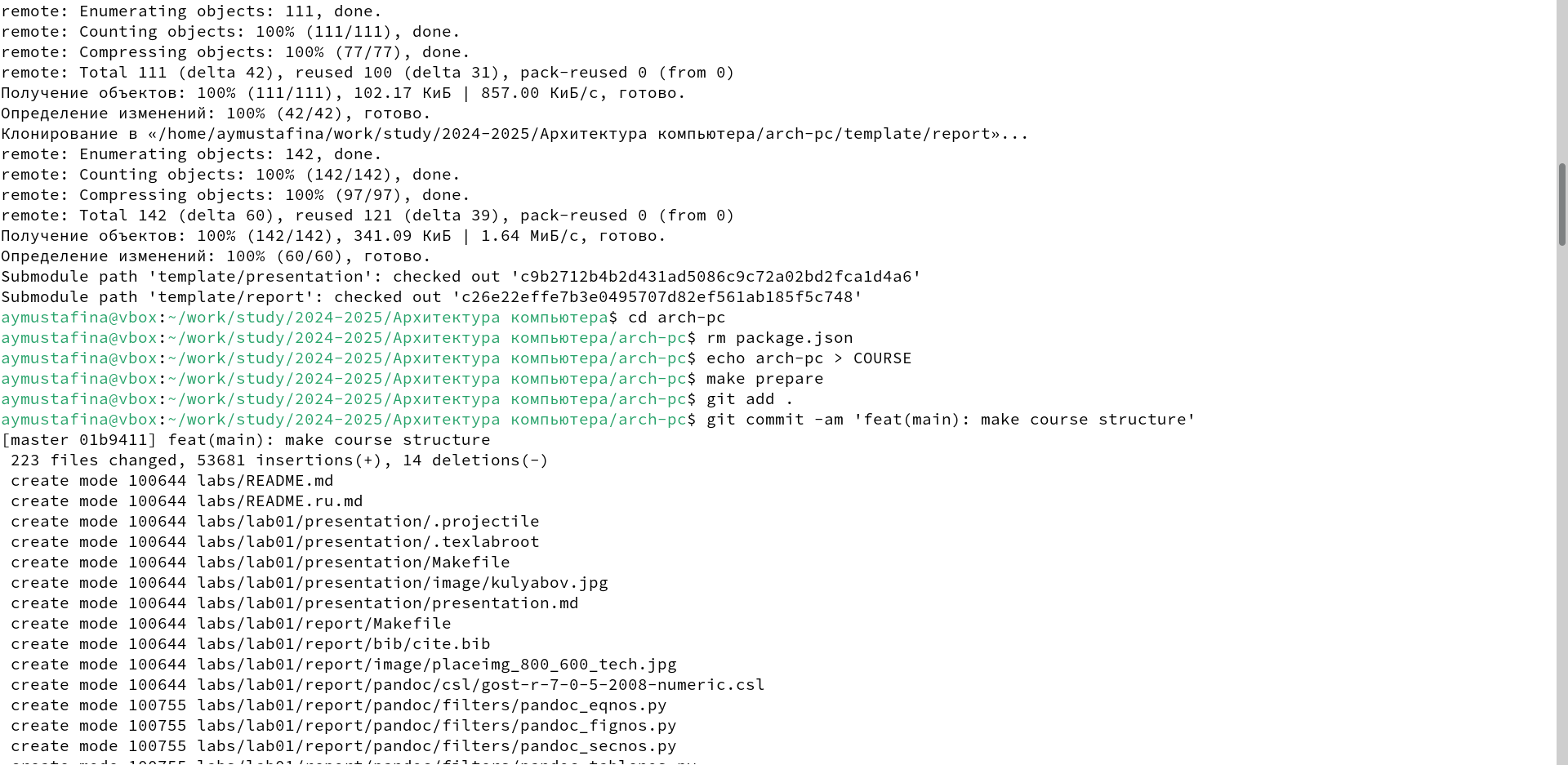


Рис. 4.6.18 Изменение и комментирование курса.

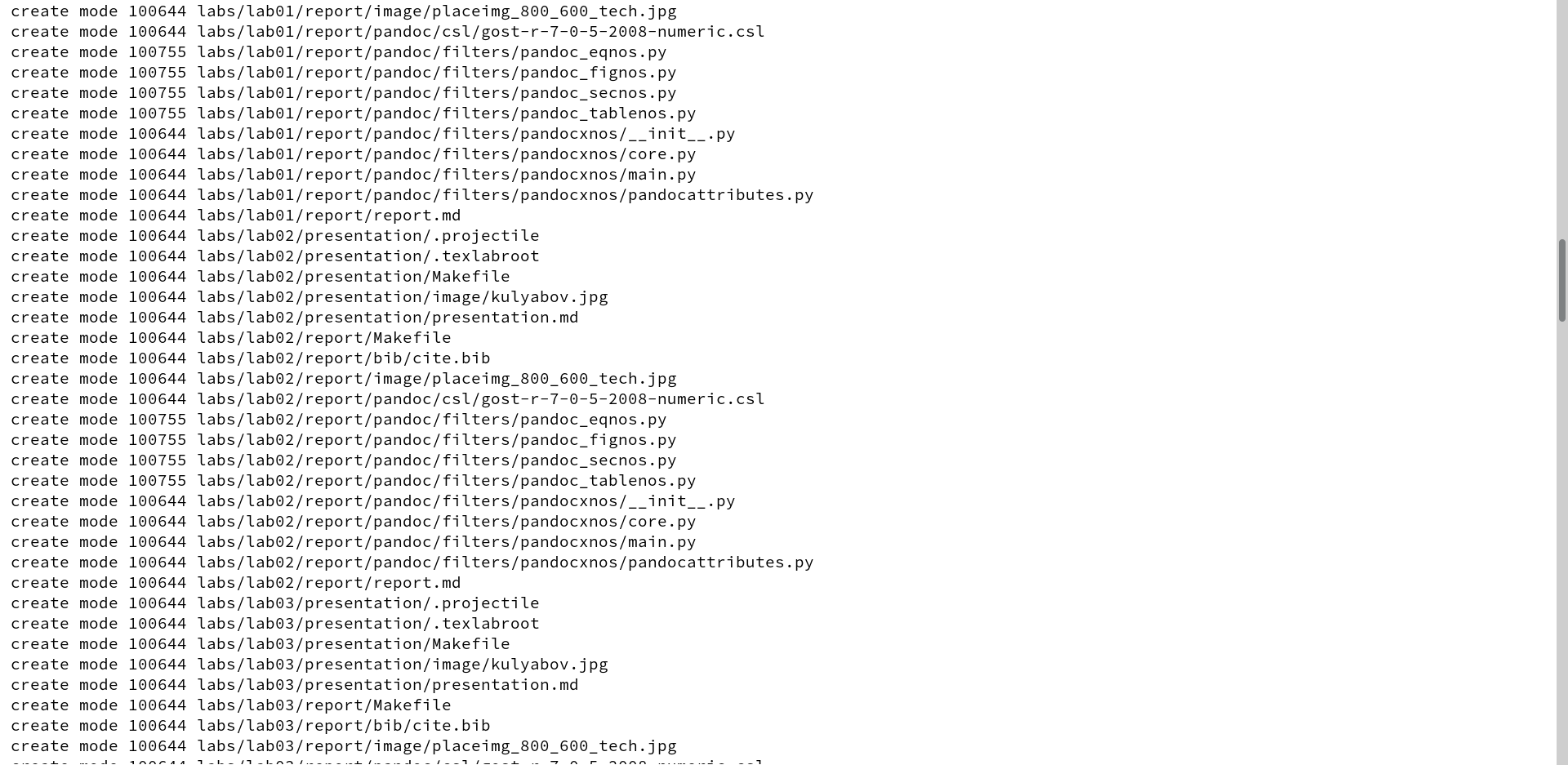


Рис. 4.6.19 Изменение и комментирование курса.

С помощью команды push отправляю все на сервер.

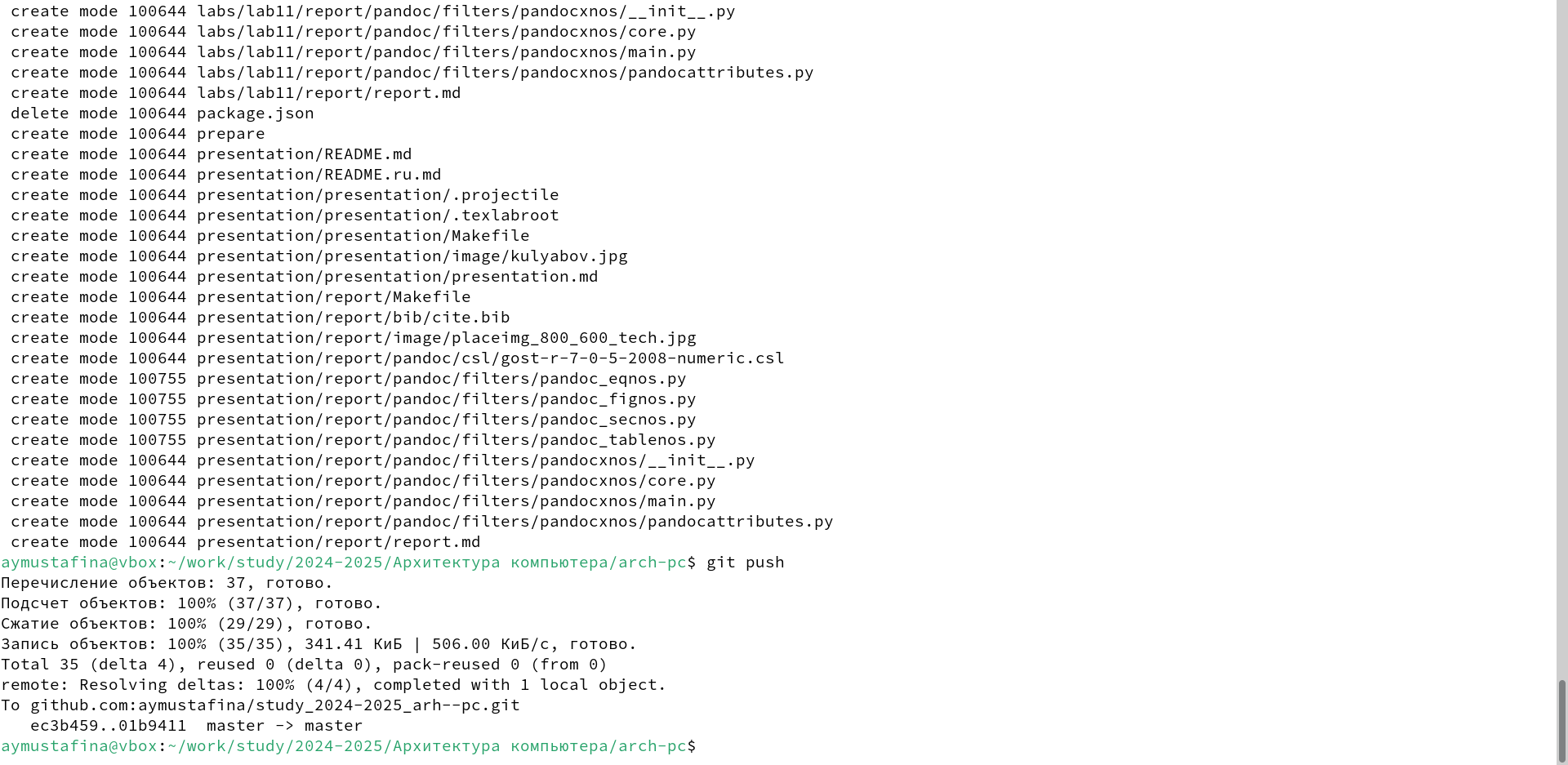


Рис. 4.6.20 Отправка файлов на сервер

Проверяю правильность выполнения работы на сайте в своем аккаунте.

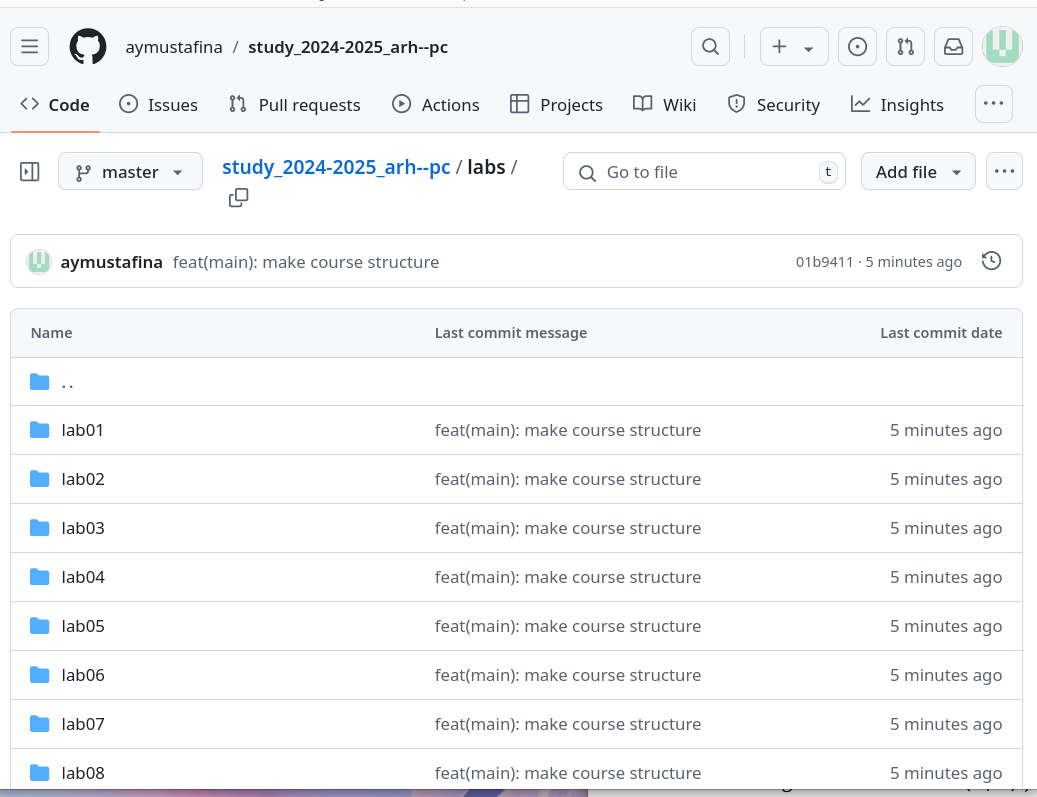


Рис. 4.6.21 Проверка правильности работы

*2.5. Задание для самостоятельной работы*

Открою каталог загрузки и увижу в нем файл с отчетом первой лабораторной работы «Л01\_Мустафина\_отчет.pdf».

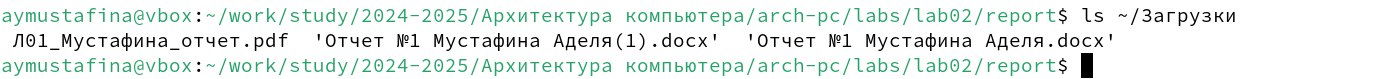
**

Рис. 5.22 Файл отчета

Посмотрю что находится в директории, далее скопирую файл с отчетом из «Загрузки» и добавлю его в данную директорию.



Рис. 5.23 Копирование файла

Прокомментирую файл на гитхабе с помощью команды «git commit –m».

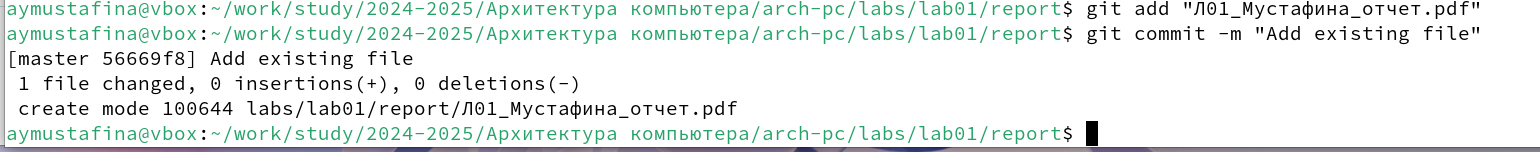


Рис. 5.24 Комментирование

Создам файл «Л02\_Мустафина\_отчет» с помощью команды touch.

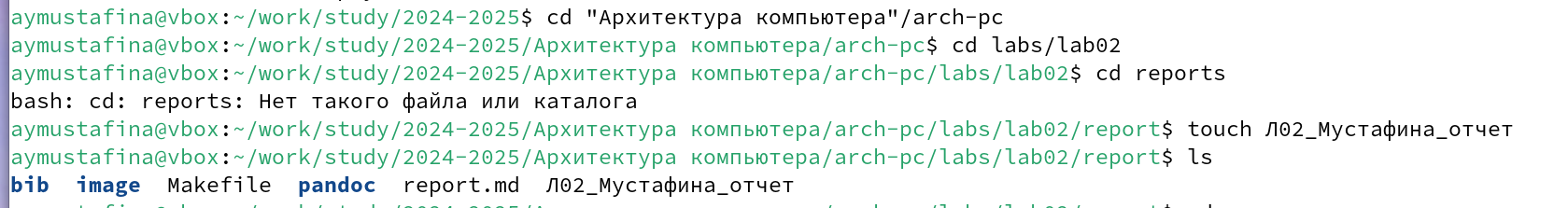


Рис. 5.25 Создание файла

Сделаю все то же самое, что делала для первого отчета.

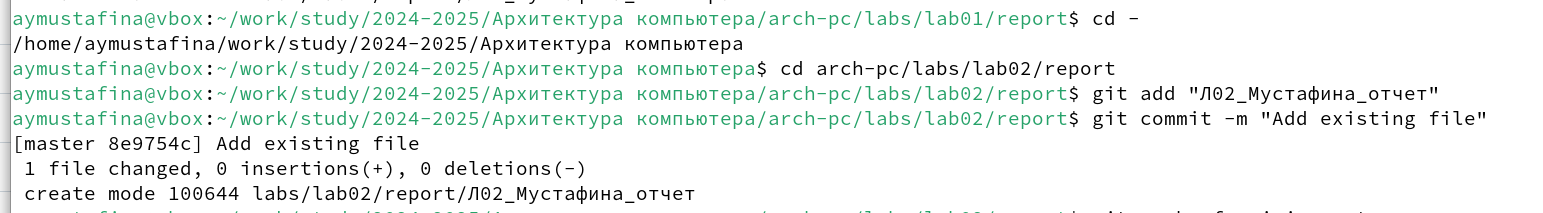


Рис. 5.26 Комментирование второго отчета

Закреплю все сделанное с помощью команды git push, отправлю все на центральный репозиторий.

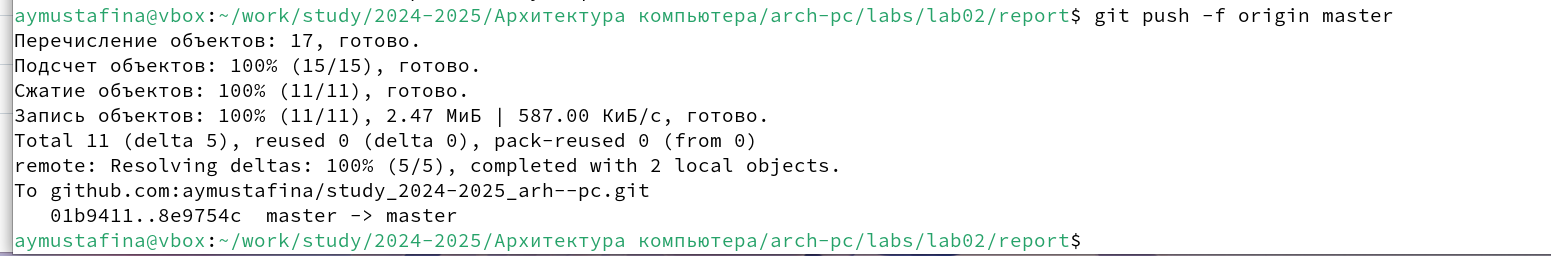
**

Рис. 5. 27 Отправка

Проверяю правильность выполнения самостоятельной работы на своем аккаунте Githab.

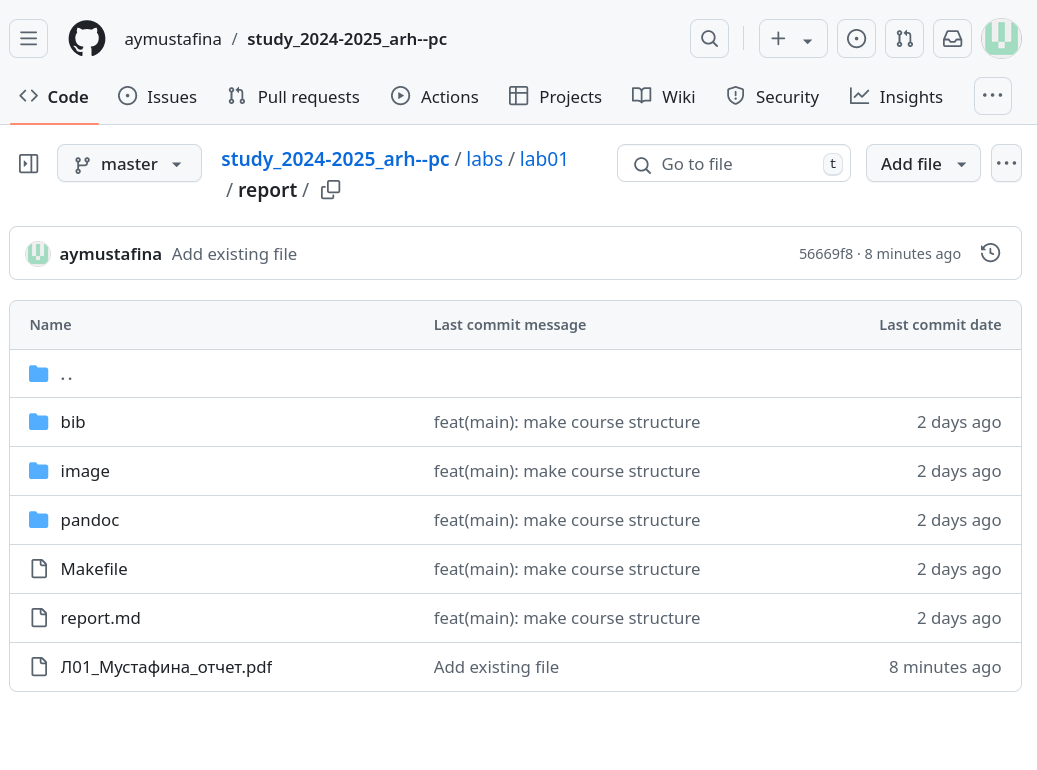


Рис. 5.28 Проверка lab01

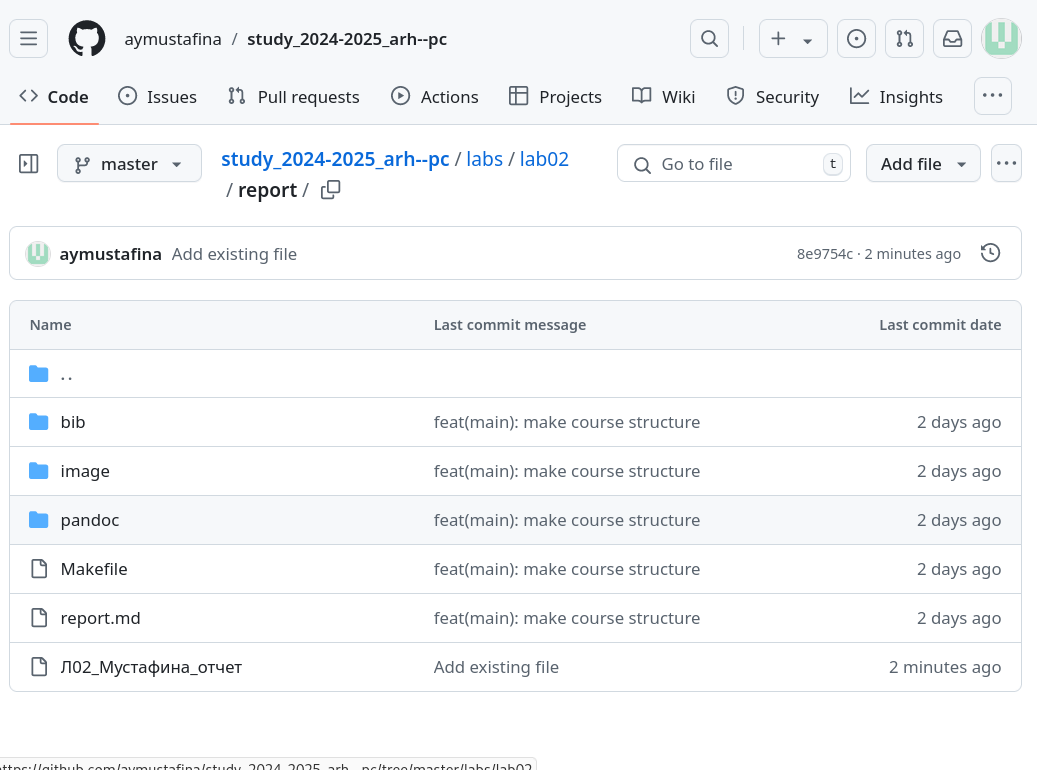


Рис. 5.29 Проверка lab02

# Выводы

По итогам выполнения лабораторной работы я научилась работать с системой git.

Научилась применять системы контроля и создавать ключи.