**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: Армихос Гонзалез Карла

Группа: НКАбд-02-24

**МОСКВА**

2024 г.

Содержание

[1. Цель работы 4](#_dy1ov8g3hbtm)

[2. Задание 5](#_jcca5ugyfw00)

[3. Теоретическое задание 6](#_dbrvr9gtas3t)

[4. Порядок выполнение лабораторной работы 8](#_38sbg4x894mw)

[4.1 Базовая настройка git 8](#_nc3dlzul0i0u)

[4.2 Создание SSH ключа 8](#_71rve66rfj5s)

[4.3 Сознание рабочего пространства и репозитория курса на основе шаблона 10](#_6v1th9xp2mt1)

[4.4 Настройка каталог курса 12](#_acrjeiycb3js)

[5. Задание для самостоятельной работы 13](#_5huc6ctmupbg)

[6. Список литературы 14](#_yk7hy2ohkeh8)

Список таблиц

[рис. 4.1.1 Настройка Git 8](#_mua8yjvwdgzm)

[рис. 4.1.2 Настроим utf-8 8](#_bt4548wzx0ta)

[рис. 4.1.3 Конфигурация 8](#_kkyx37dauax2)

[рис. 4.2.1 Создание SSH ключа 9](#_2d5hmjk97edd)

[рис. 4.2.2 Добавить созданный ключ 9](#_5yeki5scnlhd)

[рис. 4.3.1 Создать каталог 10](#_3bbnnz3qcpj9)

[рис. 4.3.2 Создать новый репозиторий 10](#_3udicofsslgk)

[рис. 4.3.3Ключ SHH 11](#_uw6ndqs4w3vr)

[рис. 4.3.4 Клонируйте созданный репозиторий 12](#_780ufaaq790n)

[рис. 4.4.1 Удалить package.json 12](#_k2b7k09dl3bm)

[рис 4.4.2 COURSE 12](#_sv06oxjoowq5)

[рис. 4.4.3 make 12](#_9588z55dupk4)

[рис. 4.4.4 Отправьте файлы 12](#_trv45opwb575)

[рис. 5.1 labs>lab2>report 13](#_ytrdljbbq51o)

[рис. 5.2 Загрузите файлы на github 13](#_ld9ynx1cm0yg)

# 1. Цель работы

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

# **2. Задание**

* Настроить Git.
* Создайте открытый ключ и загрузите его в Git.
* Создайте репозиторий и используйте его ключ
* Создайте каталог с иерархией, в котором укажите наш учебный год и предмет.
* Загружать наши отчеты в созданный репозиторий

# 3. Теоретическое задание

Системы контроля версий (VCS) — это инструменты, предназначенные для управления изменениями в файлах и документах. Они позволяют отслеживать изменения в коде, координировать работу нескольких разработчиков, а также обеспечивают возможность восстановления предыдущих версий файлов. VCS особенно полезны в программной разработке, где требуется постоянное обновление и модификация кода.

Основные понятия, связанные с VCS, включают хранилище, commit, историю и рабочую копию. *Хранилище* — это место, где сохраняются все версии проекта. **Commit**— это зафиксированное изменение в коде, которое сохраняет состояние файлов в хранилище. **История**— это последовательность всех коммитов, которая позволяет отслеживать изменения и возвращаться к предыдущим версиям. **Рабочая** **копия**— это локальная версия проекта, с которой разработчик работает, внося изменения перед их фиксацией в хранилище.

Системы контроля версий можно разделить на централизованные и децентрализованные. В централизованных VCS, таких, как SVN (Subversion), все изменения хранятся на центральном сервере, и разработчики работают с единственной версией проекта. В децентрализованных VCS, таких, как Git, каждый разработчик имеет свою полную копию репозитория, что позволяет работать автономно и синхронизировать изменения позже.

При единоличной работе с хранилищем VCS разработчик вносит изменения в рабочую копию, выполняет команду *commit*, чтобы сохранить эти изменения в хранилище, и, если необходимо, просматривает историю изменений для отслеживания прогресса или возврата к предыдущим версиям.

При работе с общим хранилищем VCS порядок действий включает клонирование репозитория, создание веток для новых функций или исправлений, внесение изменений в рабочую копию, выполнение коммитов и отправку изменений в центральное хранилище. Также важна синхронизация изменений с другими разработчиками, чтобы избежать конфликтов.

Основные задачи, решаемые инструментальным средством Git, включают управление версиями, совместная работа над проектами, ведение истории изменений и возможность отката к предыдущим состояниям кода. Git также поддерживает ветвление и слияние, что позволяет параллельно разрабатывать новые функции.

Некоторые основные команды Git включают:

- git init: создание нового репозитория.

- git clone: копирование удаленного репозитория на локальную машину.

- git add: добавление изменений в индекс перед коммитом.

- git commit: фиксация изменений в хранилище.

- git push: отправка локальных изменений в удалённый репозиторий.

- git pull: получение изменений из удалённого репозитория.

- git branch: управление ветками.

Примеры использования Git при работе с локальными и удалёнными репозиториями включают:

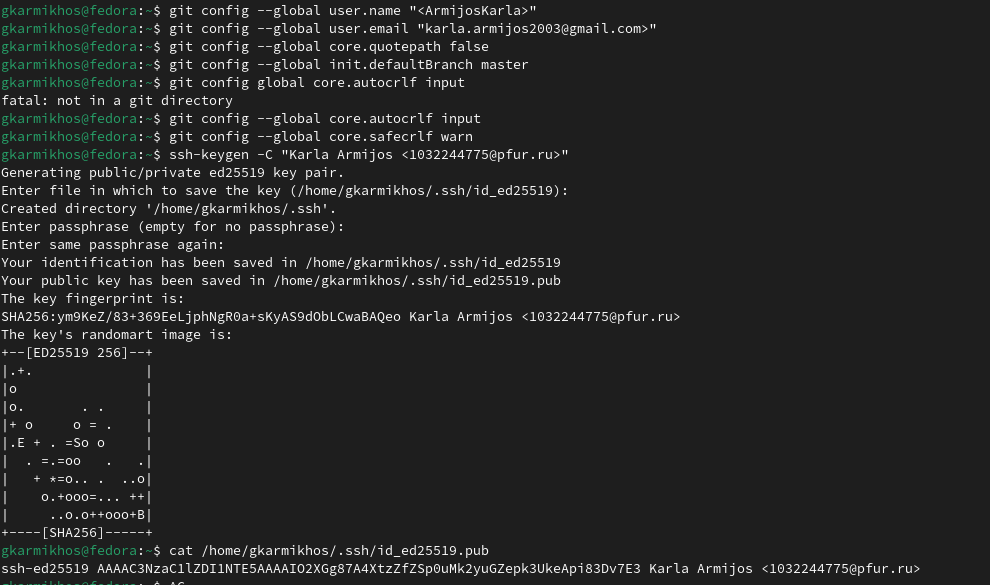
- Для локального репозитория: git init, затем (git add .) и git commit -m "Первый коммит" для сохранения изменений.

- Для удалённого репозитория: git clone <URL>, чтобы скопировать проект, после чего можно вносить изменения, а затем использовать git push для отправки своих коммитов на сервер.

# 4. Порядок выполнение лабораторной работы

## 4.1 Базовая настройка git

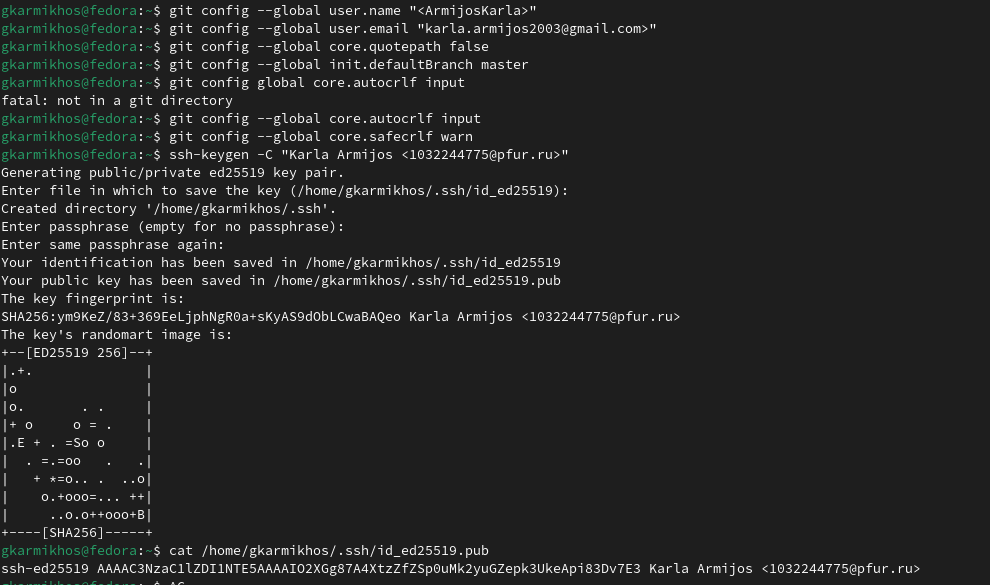
Мы открываем терминал и вводим следующие команды 1) git config--global user.name "<Имя, фамилия>" 2) git config--global user. email "<work@mail>", указав имя и адрес электронной почты владельца репозитория (рис. 4.1.1)



### рис. 4.1.1 Настройка Git

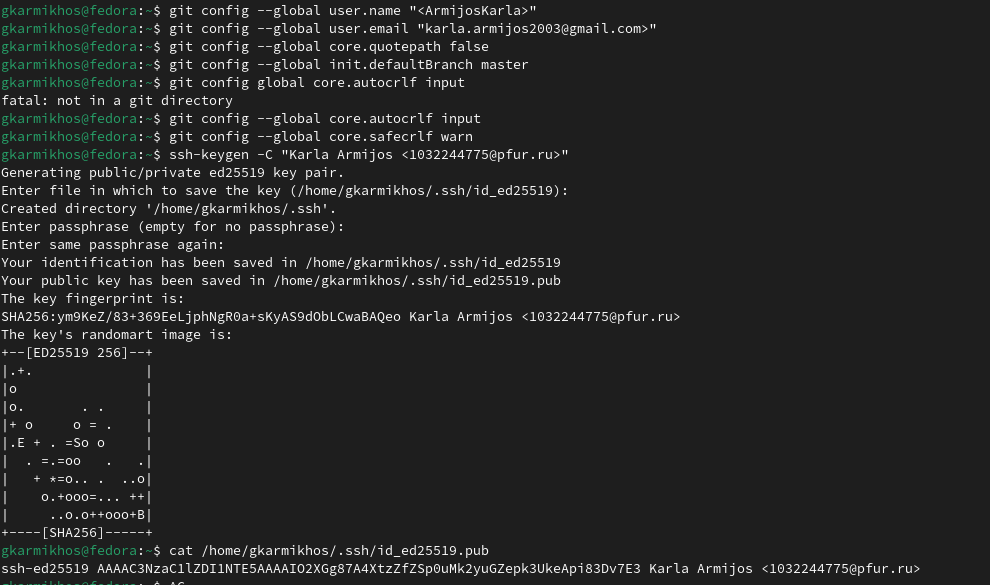
### 

Настроим utf-8 в выводе сообщений git (рис. 4.1.2)



### рис. 4.1.2 Настроим utf-8

Зададим имя начальной ветки, Параметр autocrlf, Параметр safecrlf (рис. 4.1.3)

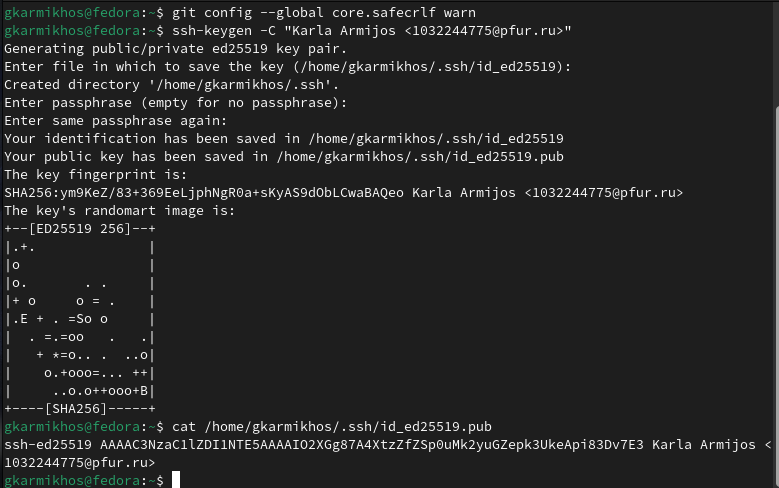


### рис. 4.1.3 Конфигурация

### 

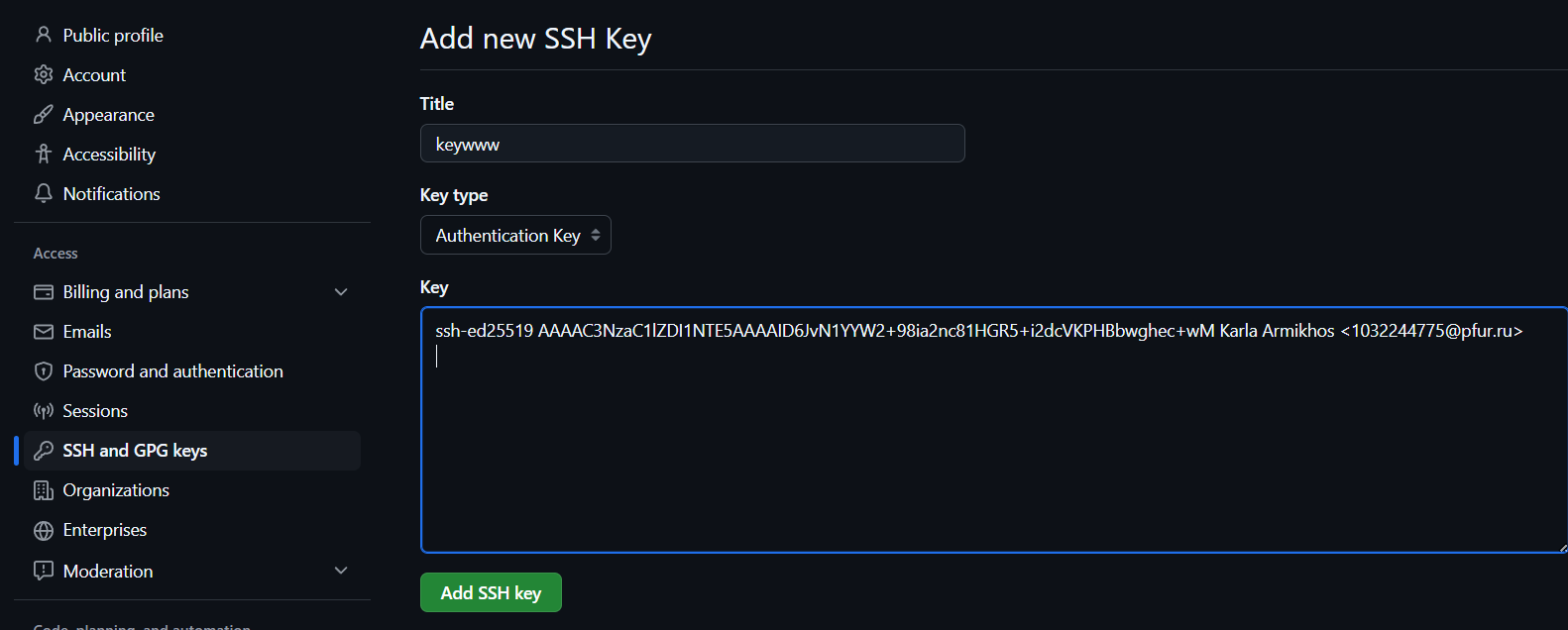
## 4.2 Создание SSH ключа

Создать открытый ключ с помощью команды ssh-keygen-C " Имяя <работа@почта>" (рис. 4.2.1)



### рис. 4.2.1 Создание SSH ключа

Мы копируем созданный нами ключ в Git, сначала нам нужно войти в конфигурацию Git, а в разделе New SSH key мы вставляем наш ключ, добавляем заголовок (рис. 4.2.2)

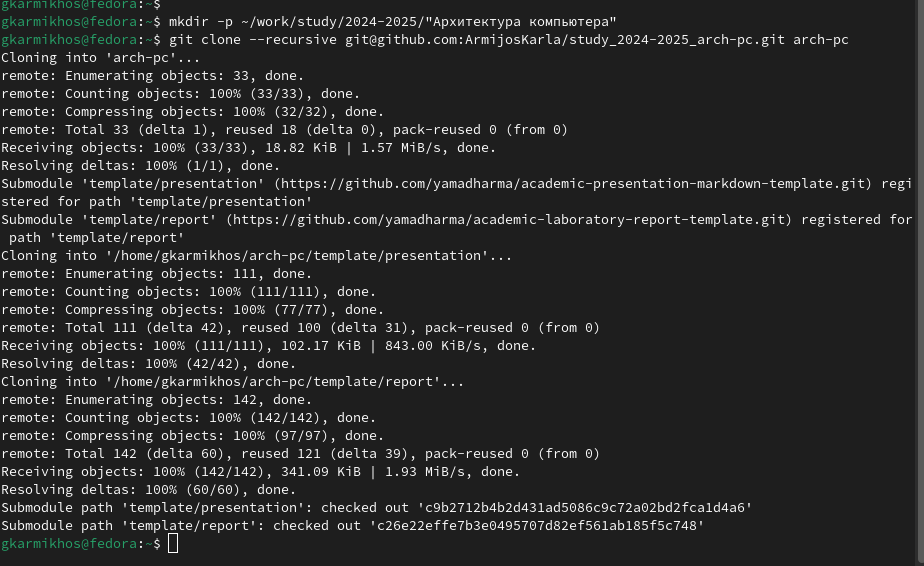


### рис. 4.2.2 Добавить созданный ключ

### 

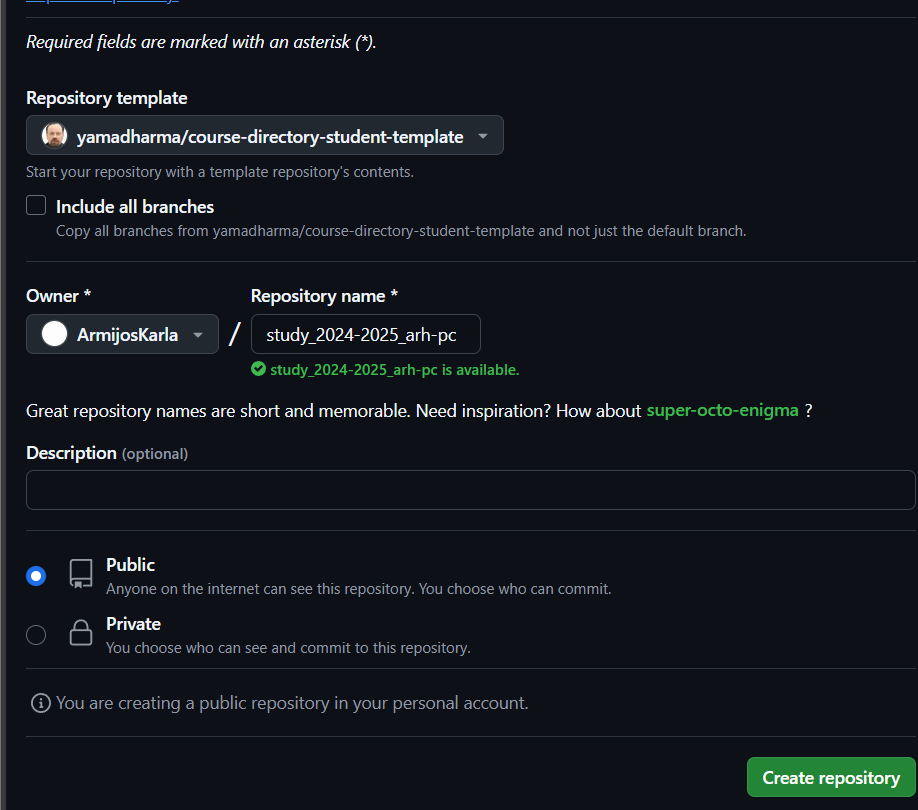
## 4.3 Сознание рабочего пространства и репозитория курса на основе шаблона

Создать каталог с разными иерархиями mkdir -p ~/work/study/2024-2025/"Архитектура компьютера" (рис. 4.3.1)



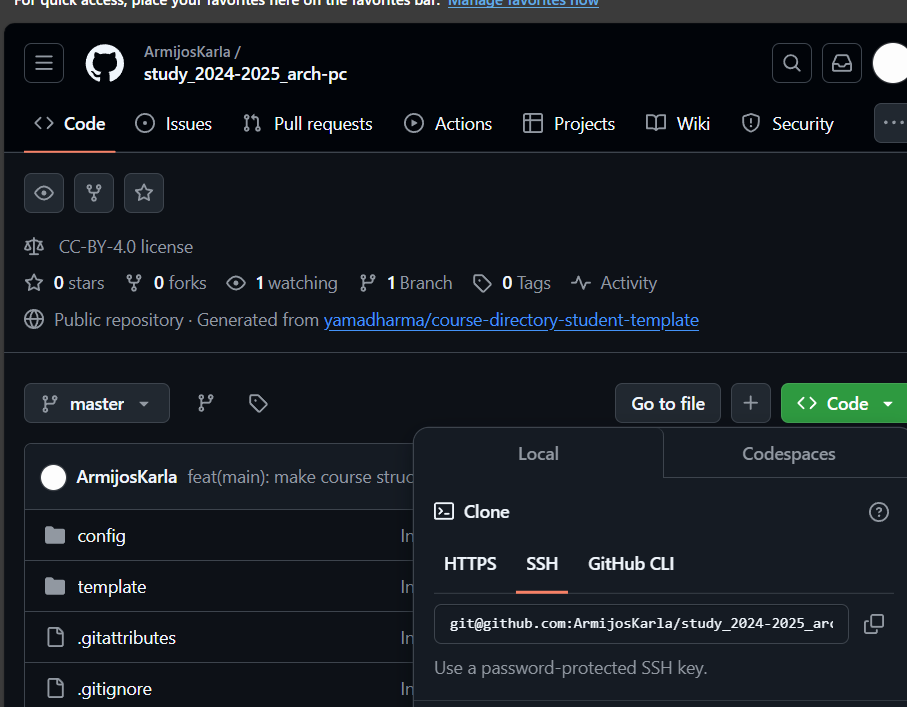
### рис. 4.3.1 Создать каталог

Создайте репозиторий на основе шаблона через веб-интерфейс github, задайте имя репозитория (рис. 4.3.2)



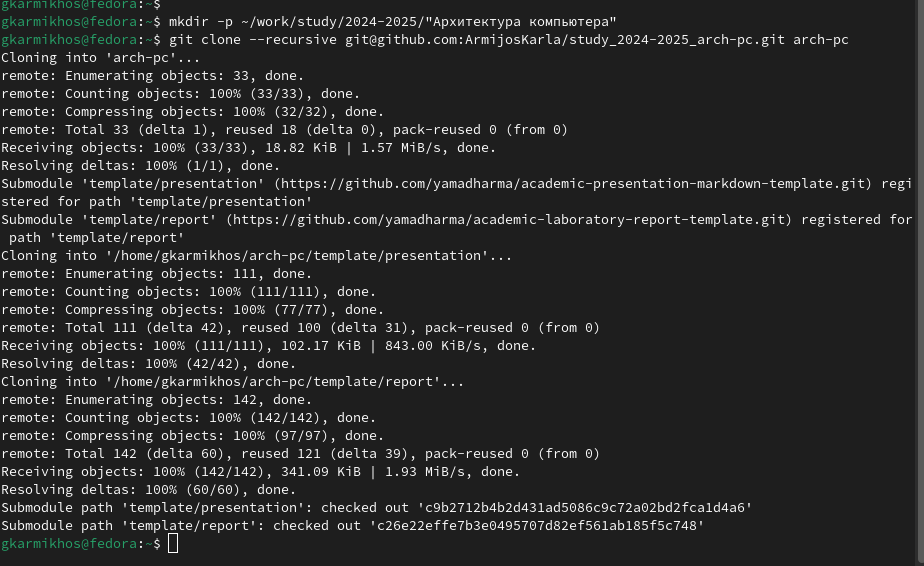
### рис. 4.3.2 Создать новый репозиторий

Ключ SHH из нашего репозитория (рис. 4.3.3)



### рис. 4.3.3Ключ SHH

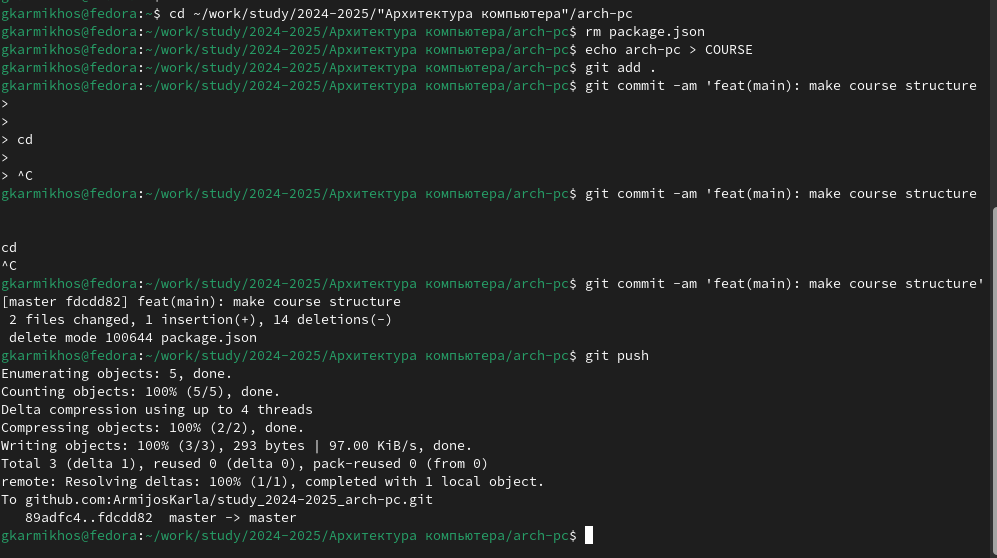
Клонировать папку с помощью сгенерированного ключа нашего репозитория (рис. 4.3.4)



### рис. 4.3.4 Клонируйте созданный репозиторий

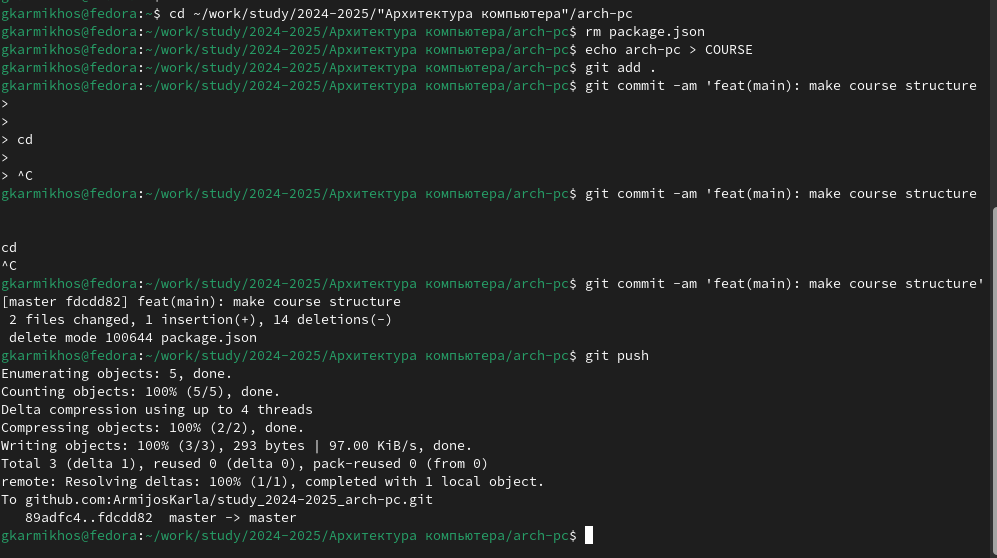
## 4.4 Настройка каталог курса

Перейдите в каталог курса и удалите лишние файлы (рис. 4.4.1)



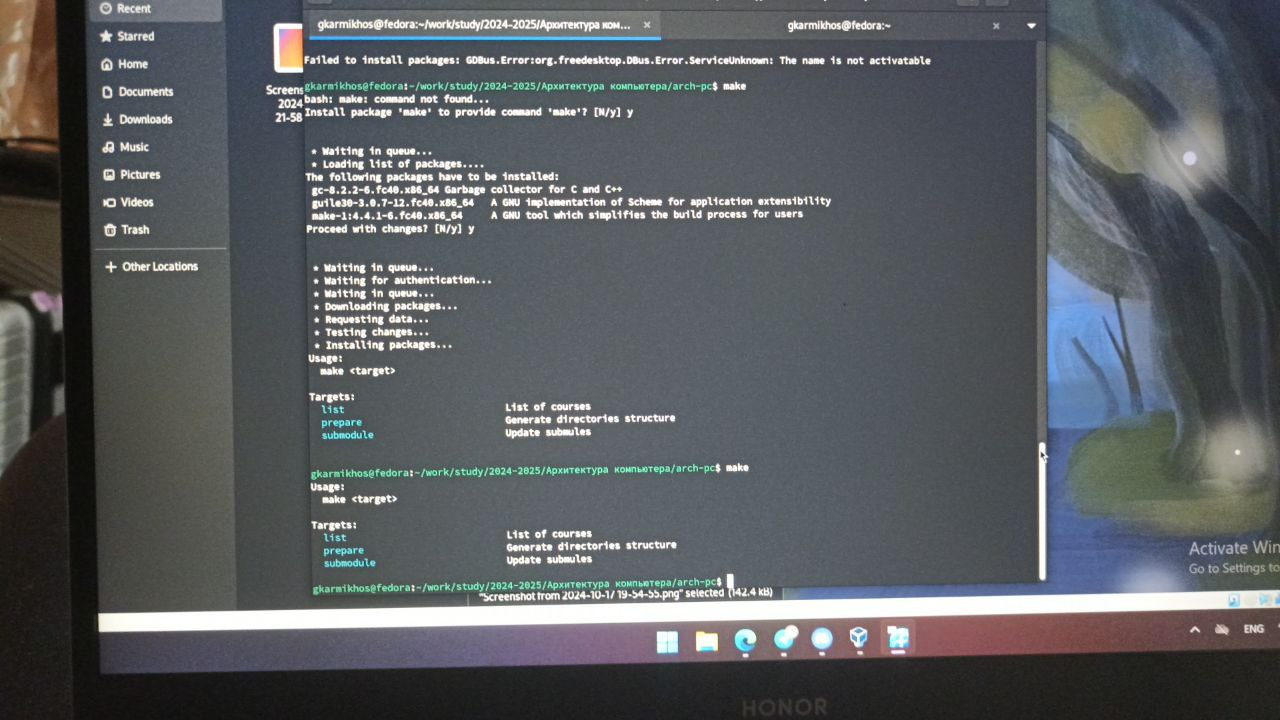
### рис. 4.4.1 Удалить package.json

Создайте необходимые каталоги command course (рис 4.4.2)



### рис 4.4.2 COURSE

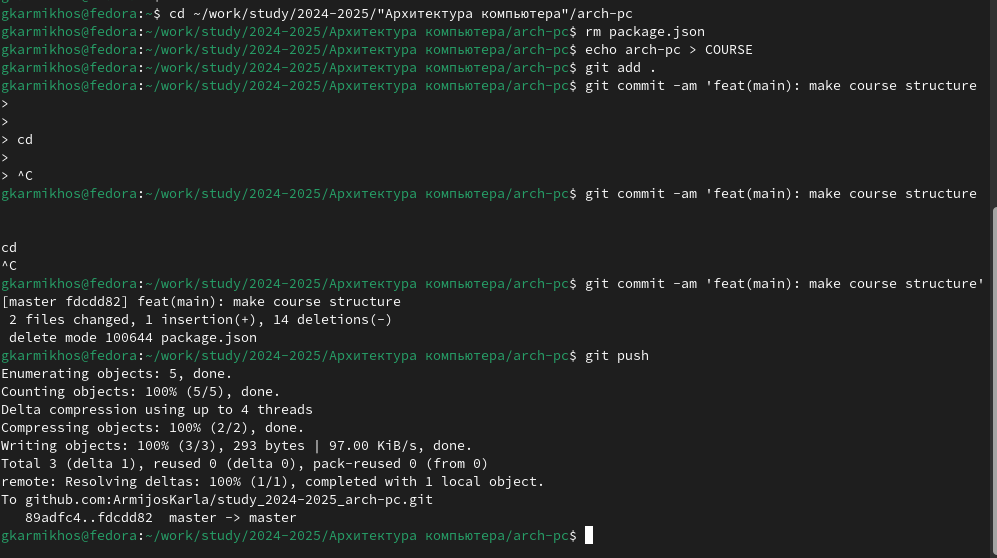
Команд make (рис. 4.4.3)

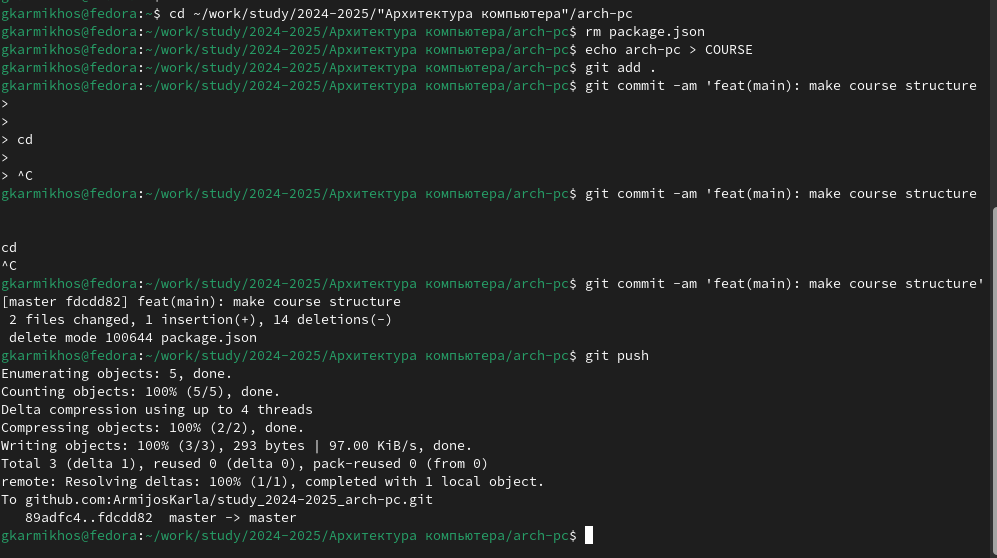


### рис. 4.4.3 make

Отправьте файлы на сервер: помощью команды git add . / f=git commit y git push

(рис. 4.4.4)

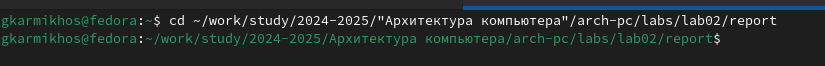




### рис. 4.4.4 Отправьте файлы

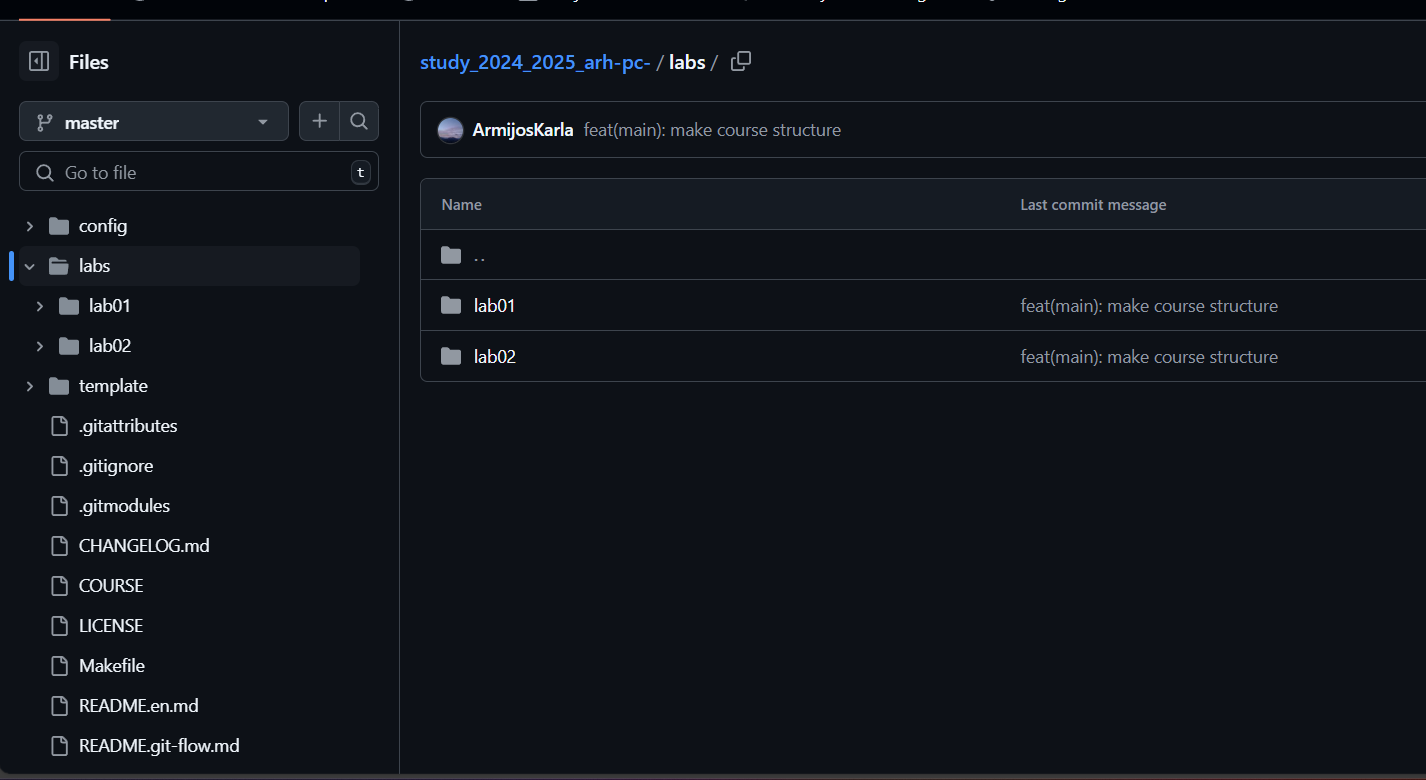
# 5. Задание для самостоятельной работы

Создайте отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs>lab02>report) (рис. 5.1).



### рис. 5.1 labs>lab2>report

Загрузите файлы на github (рис 5.2)



### рис. 5.2 Загрузите файлы на github

# 6. Список литературы

[Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/2089082/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%962.%20%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8F%20%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B9%20Git.pdf)