**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**по лабораторной работе № 2**

*дисциплина: Архитектура компьютеров*

*и операционных систем*

Студент: Акмурадов Тимур

Группа: НБИбд-02-23

**МОСКВА**

2023 г

**Содержание**

1. [Цель работы](#_bookmark0) 5
2. [Задание](#_bookmark1) 6
3. [Теоретическое введение](#_bookmark2) 7
4. [Выполнение лабораторной работы](#_bookmark3) 9

[4.1 Настройка GitHub](#_bookmark4) . . . . . . . . . . . . . . . . . . . . . . . . . . 9

* 1. [Базовая настройка Git](#_bookmark7) 10
  2. [Создание SSH-ключа](#_bookmark13) 11
  3. [Создание рабочего пространства и репозитория курса на основе](#_bookmark19) [шаблона](#_bookmark19) 14
  4. [Создание репозитория курса на основе шаблона](#_bookmark21) 15
  5. [Настройка каталога курса](#_bookmark28) 17
  6. [Выполнение заданий для самостоятельной работы](#_bookmark35) 20

1. [Выводы](#_bookmark54) 27
2. [Список литературы](#_bookmark55) 28

# Цель работы

Целью данной работы является изучить идеологию и применение средств контроля версий, а также 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 работ

# Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при ра- боте нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен до- ступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, про- изведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется. В классических системах контроля версий используется централизованная модель, предполагающая наличие еди- ного репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник про- екта (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 уда- ляются 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 по- следовательными версиями, что позволяет уменьшить объём хранимых данных. Системы контроля версий поддерживают возможность отслеживания и разреше- ния конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить изменения, сделанные разными участниками, вручную выбрать нужную версию, отменить изменения вовсе или заблокиро- вать файлы для изменения. В зависимости от настроек блокировка не позволяетдругим 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 измене- ний до точки ветвления версий и собственные истории изменений каждой ветви. 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 распреде- лённых — 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. Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория (при этом в локальное дерево до начала этой процедуры не должно было вноситься изменений). Затем можно вносить изменения в локальном дереве и/или ветке. После завершения внесения какого-то изменения в файлы и/или каталоги проекта необходимо разместить их в центральном репозитории

# Выполнение лабораторной работы

## Настройка GitHub

Создаю учетную запись на сайте GitHub (рис. [4.1).](#_bookmark5) Далее я заполняю основные данные учетной записи и регистрирую аккаунт.

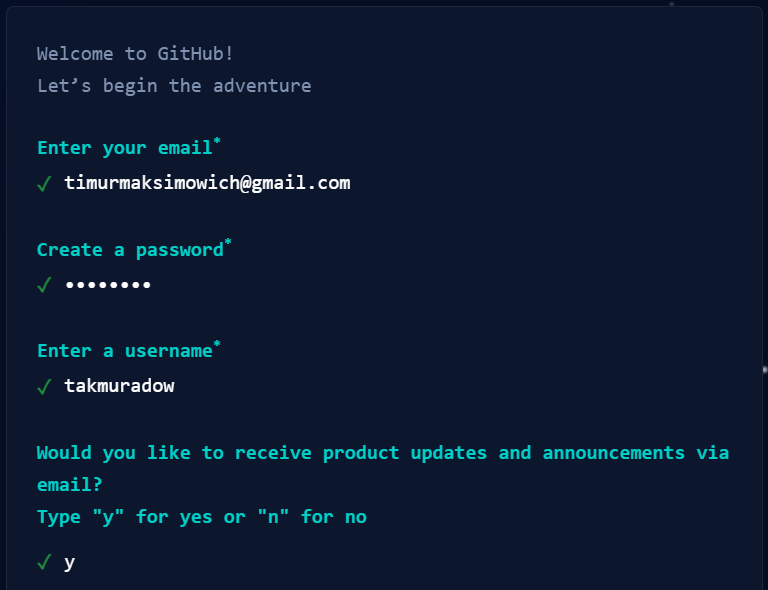


Рис. 4.1: Заполнение данных учетной записи GitHub

Аккаунт создан (рис. [4.2).](#_bookmark6)

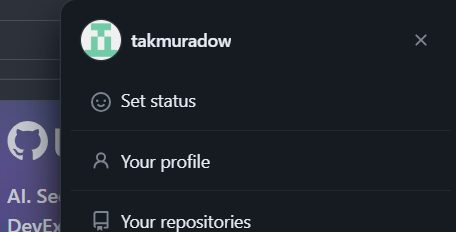


Рис. 4.2: Аккаунт GitHub

## Базовая настройка Git

Запускаю виртуальную машину, затем в терминале задаю предварительную конфигурацию git. Ввожу команду git config –global user.name “”, указывая свое имя и команду git config –global user.email “work@mail”, указывая в ней электронную почту владельца, то есть мою (рис. [4.3).](#_bookmark8)

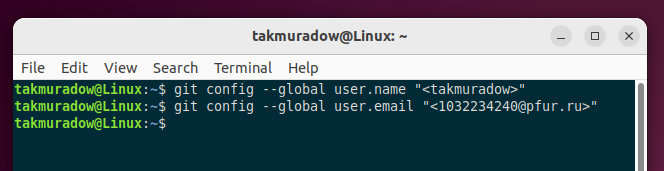


Рис. 4.3: Предварительная конфигурация git

Настраиваю utf-8 в выводе сообщений git для корректного отображения символов (рис. [4.4).](#_bookmark9)



Рис. 4.4: Настройка кодировки

Задаю имя «master» для начальной ветки (рис. [4.5).](#_bookmark10)



Рис. 4.5: Создание имени для начальной ветки

Задаю параметр autocrlf со значением input, так как я работаю в системе Linux, чтобы конвертировать CRLF в LF только при коммитах (рис. [4.6).](#_bookmark11) CR и LF – это символы, которые можно использовать для обозначения разрыва строки в текстовых файлах.



Рис. 4.6: Параметр autocrlf

Задаю параметр safecrlf со значением warn, так Git будет проверять преобразование на обратимость (рис. [4.7).](#_bookmark12) При значении warn Git только выведет предупреждение, но будет принимать необратимые конвертации.



Рис. 4.7: Параметр safecrlf

## Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый). Для этого ввожу команду ssh-keygen -C “Имя Фамилия, work@email”, указывая имя владельца и электронную почту владельца (рис. [4.8).](#_bookmark14) Ключ автоматически сохранится в каталоге ~/.ssh/.

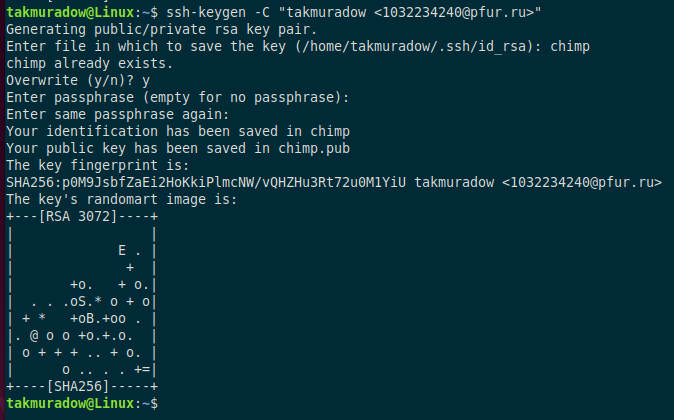


Рис. 4.8: Генерация SSH-ключа

Xclip – утилита, позволяющая скопировать любой текст через терминал. Оказывается, в дистрибутиве Linux Ubuntu ее сначала надо установить. Устанавливаю xclip с помощью команды apt-get install с ключом -y от имени суперпользователя, введя в начале команды sudo (рис. [4.9).](#_bookmark15)

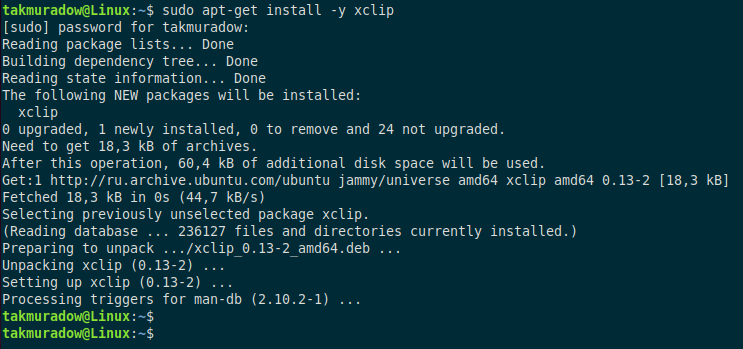


Рис. 4.9: Установка утилиты xclip

Копирую открытый ключ из директории, в которой он был сохранен, с помощью утилиты xclip (рис. [4.10).](#_bookmark16)



Рис. 4.10: Копирование содержимого файла

Открываю браузер, захожу на сайт GitHub. Открываю свой профиль и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key» (рис. [4.11).](#_bookmark17)

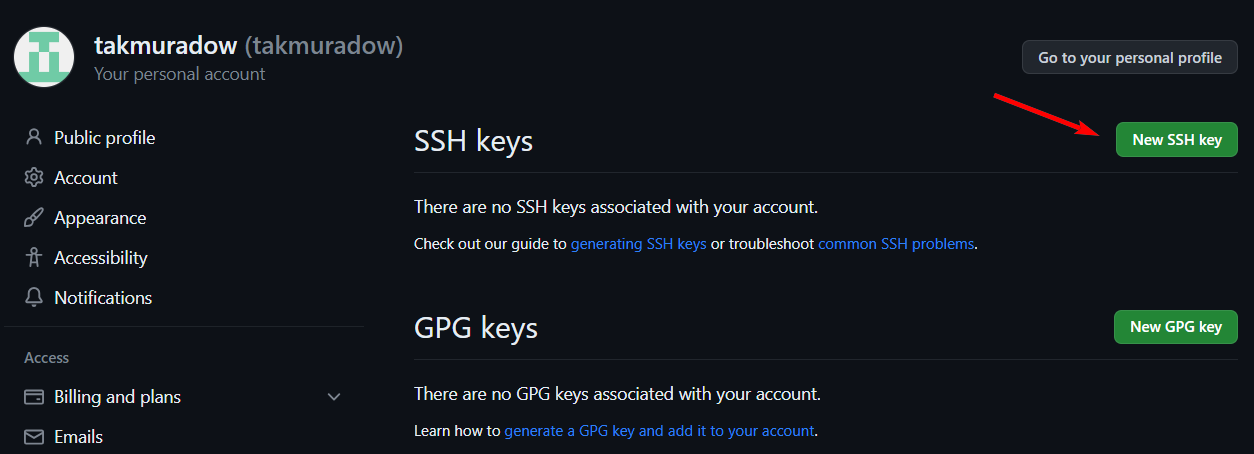


Рис. 4.11: Окно SSH and GPG keys

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

(рис. [4.12).](#_bookmark18)

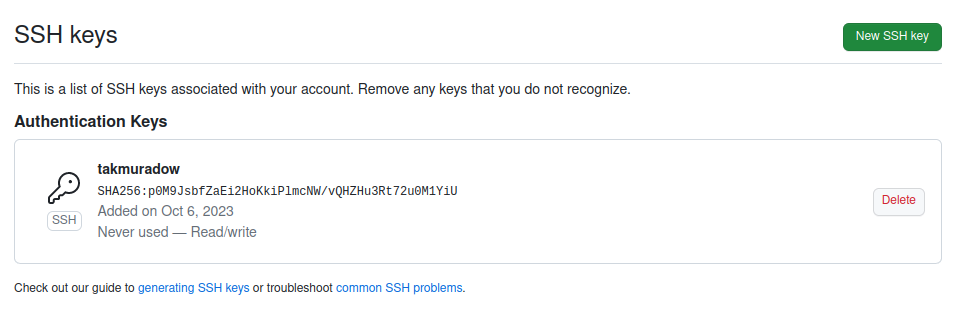


Рис. 4.12: Добавление ключа

## Создание рабочего пространства и репозитория курса на основе шаблона

Закрываю браузер, открываю терминал. Создаю директорию, рабочее пространство, с помощью утилиты mkdir, благодаря ключу -p создаю все директории после домашней ~/work/study/2023-2024/“Computer architecture” рекурсивно. Далее проверяю с помощью ls, действительно ли были созданы необходимые мне каталоги (рис. [4.13).](#_bookmark20)

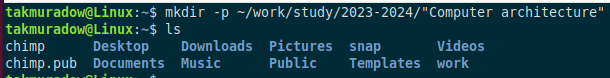


Рис. 4.13: Создание рабочего пространства

## Создание репозитория курса на основе шаблона

В браузере перехожу на страницу репозитория с шаблоном курса по адресу https://github.com/yamadharma/course-directory-student-template. Далее выбираю «Use this template», чтобы использовать этот шаблон для своего репозитория (рис. [4.14).](#_bookmark22)

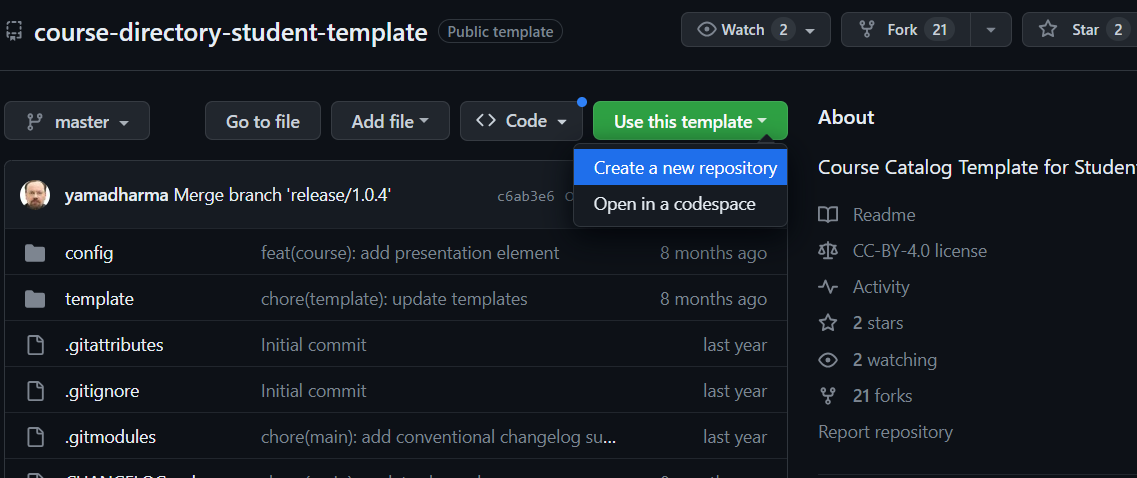


Рис. 4.14: Страница шаблона для репозитория

В открывшемся окне задаю имя репозитория (Repository name): study\_2022–2023\_arch- pc и создаю репозиторий, нажимаю на кнопку «Create repository»

(рис. [4.15).](#_bookmark23)

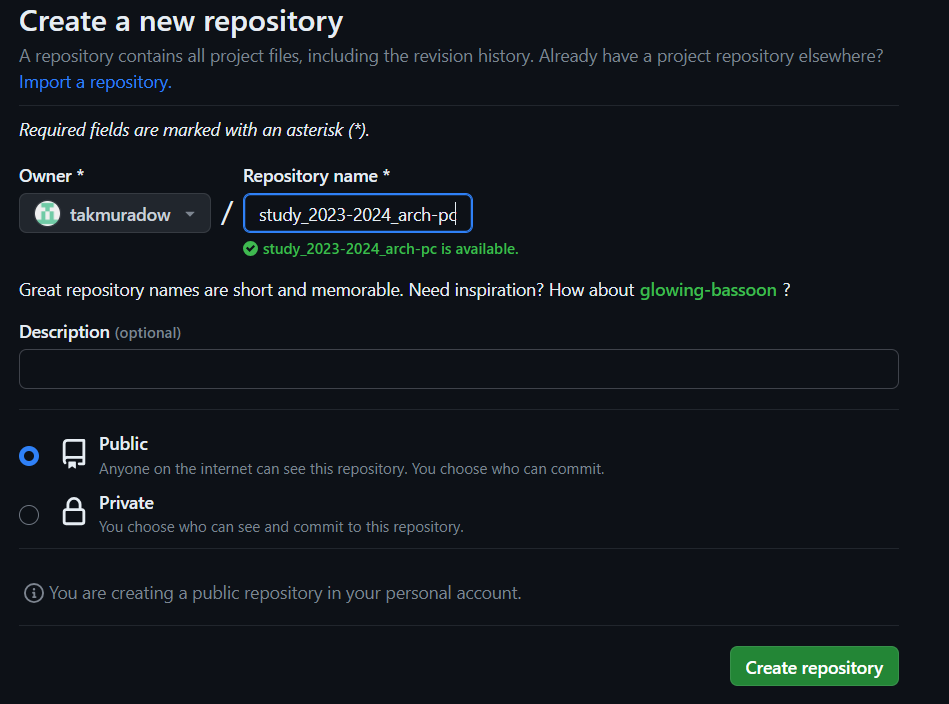


Рис. 4.15: Окно создания репозитория

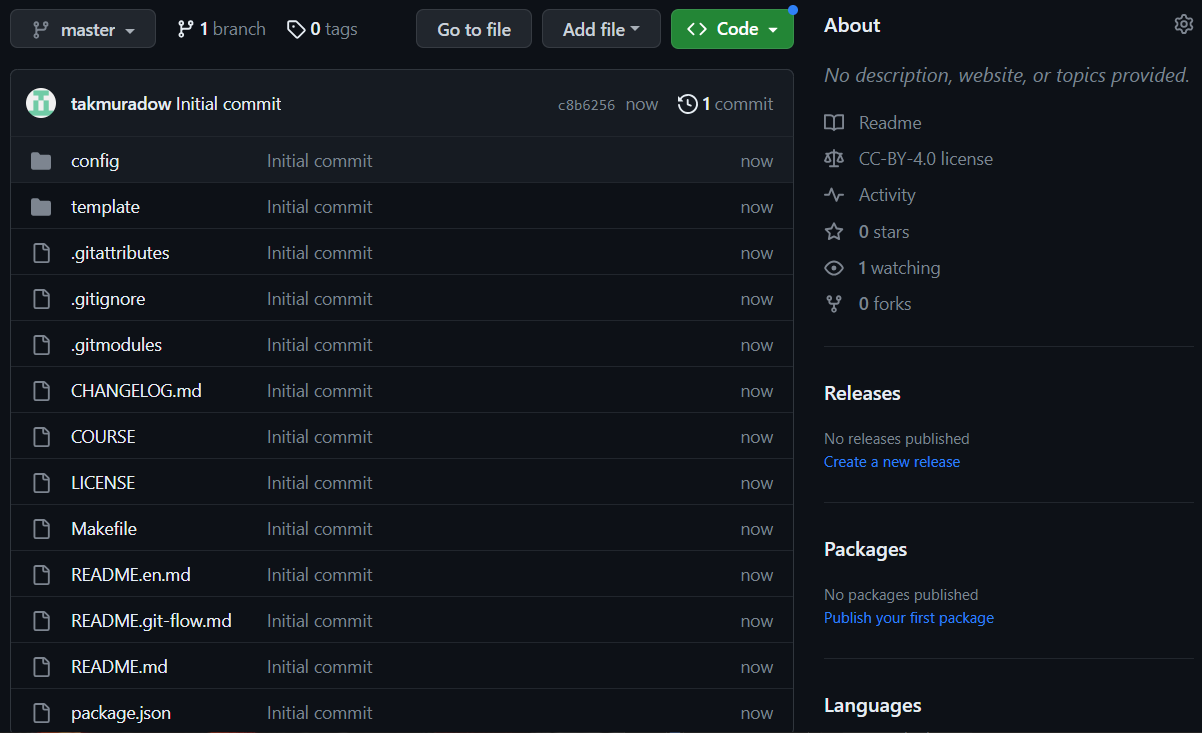


Рис. 4.16: Созданный репозиторий

Через терминал перехожу в созданный каталог курса с помощью утилиты cd (рис. [4.17).](#_bookmark25)



Рис. 4.17: Перемещение между директориями

Копирую ссылку для клонирования на странице созданного репозитория, сначала перейдя в окно «code», далее выбрав в окне вкладку «SSH» (рис. [4.18).](#_bookmark27)

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Рис. 4.18: Окно с ссылкой для копирования репозитория

Клонирую созданный репозиторий с помощью команды

git clone –recursive git@github.com:/study\_2023–2024\_arh-pc.git arch-pc (рис. [4.19).](#_bookmark26)

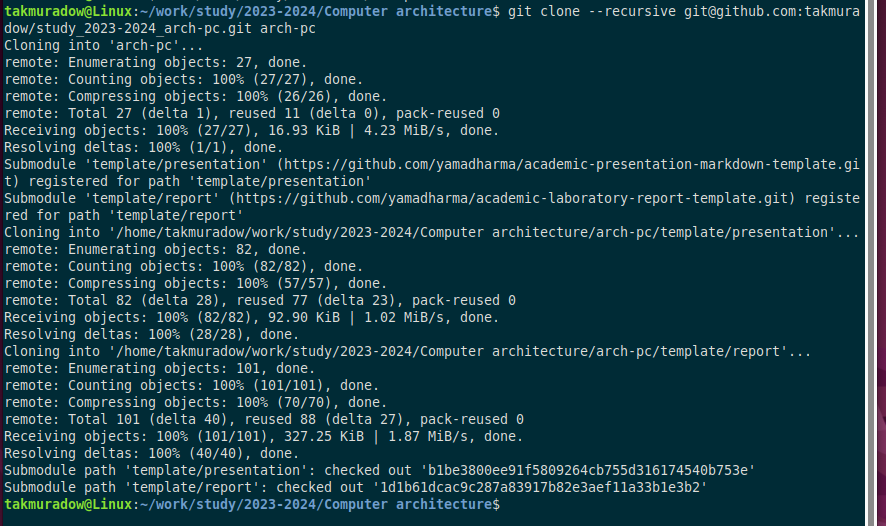


Рис. 4.19: Клонирование репозитория

## Настройка каталога курса

Перехожу в каталог arch-pc с помощью утилиты cd (рис. [4.20).](#_bookmark29)



Рис. 4.20: Перемещение между директориями

Удаляю лишние файлы с помощью утилиты rm (рис. [4.21).](#_bookmark30)



Рис. 4.21: Удаление файлов Создаю необходимые каталоги (рис. [4.22).](#_bookmark31)



Рис. 4.22: Создание каталогов

Отправляю созданные каталоги с локального репозитория на сервер: добавляю все созданные каталоги с помощью git add, комментирую и сохраняю изменения на сервере как добавление курса с помощью git commit (рис. [4.23).](#_bookmark32)

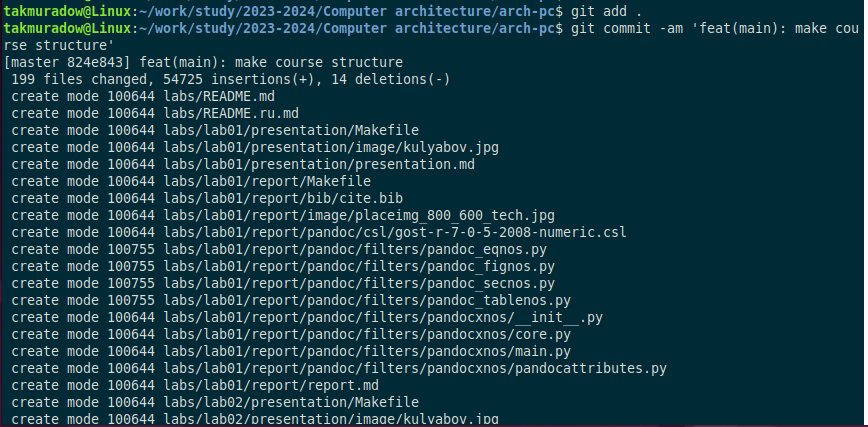


Рис. 4.23: Добавление и сохранение изменений на сервере

Отправляю все на сервер с помощью push (рис. [4.24).](#_bookmark33)

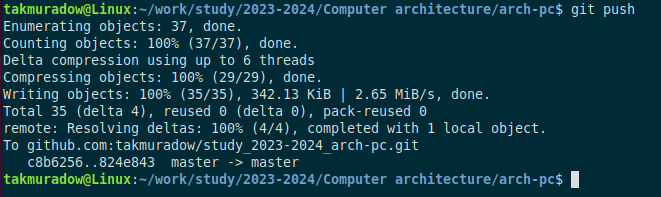


Рис. 4.24: Выгрузка изменений на сервер

Проверяю правильность выполнения работы на самом сайте GitHub (рис. [4.25).](#_bookmark34)

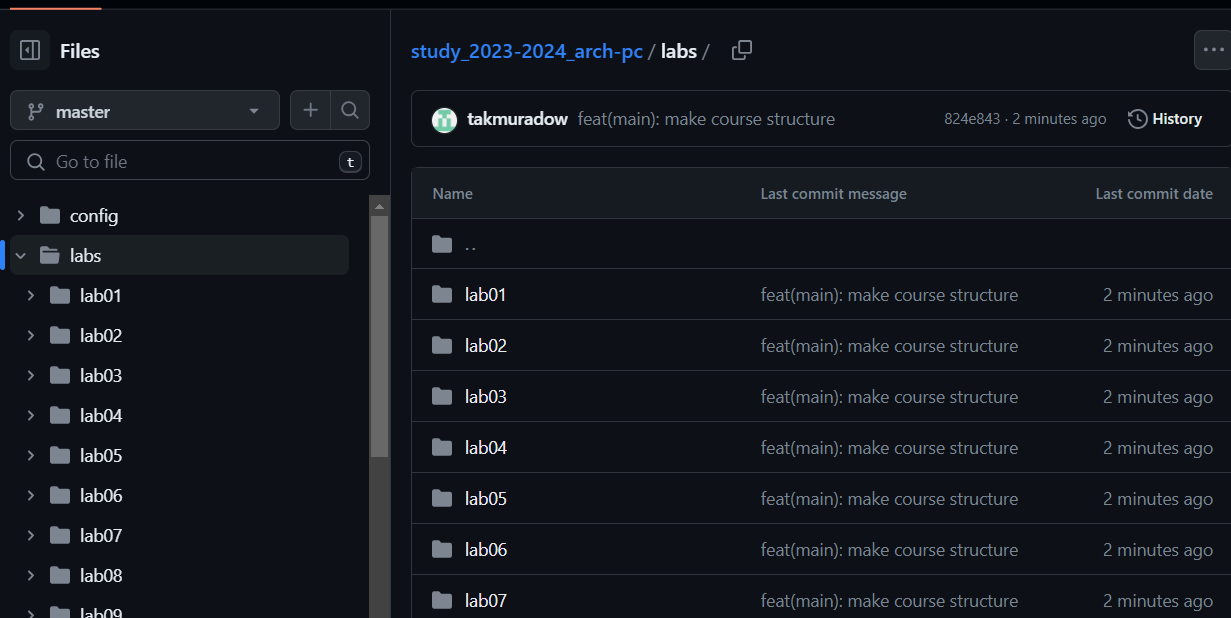


Рис. 4.25: Страница репозитория

## Выполнение заданий для самостоятельной работы

* + 1. Перехожу в директорию labs/lab02/report с помощью утилиты cd. Создаю в каталоге файл для отчета по третьей лабораторной работе с помощью утилиты touch (рис. [4.26).](#_bookmark36)



Рис. 4.26: Создание файла

Оформить отчет я смогу в текстовом процессоре LibreOffice Writer, найдя его в меню приложений (рис. [4.27).](#_bookmark37)

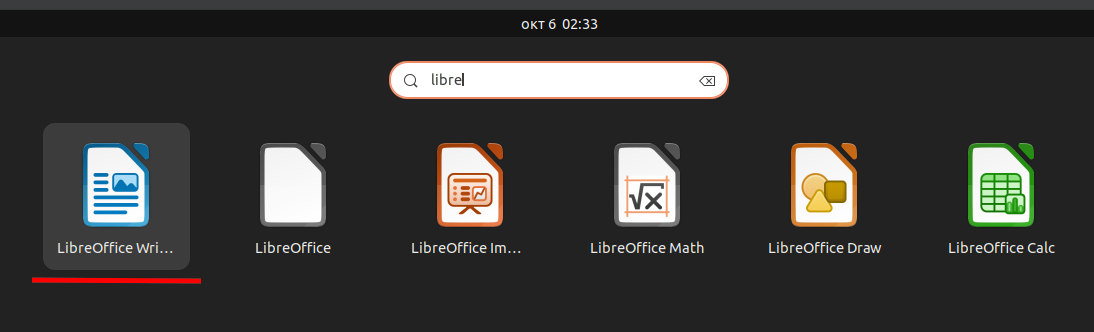


Рис. 4.27: Меню приложений

После открытия текстового редактора открываю в нем созданный файл и могу начать в нем работу над отчетом (рис. [4.28).](#_bookmark38)

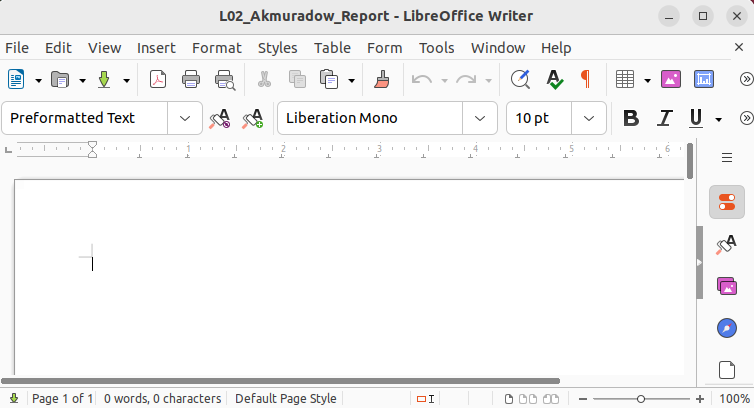


Рис. 4.28: Работа с отчетом в текстовом редакторе

* + 1. Перехожу из подкаталога lab02/report в подкаталог lab01/report с помощью утилиты cd (рис. [4.29).](#_bookmark39)

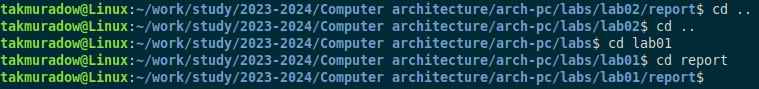


Рис. 4.29: Перемещение между директориями

Проверяю местонахождение файлов с отчетами по первой лабораторной работe. Она должна быть в подкаталоге домашней директории «Downloads», для проверки использую команду ls (рис. [4.30).](#_bookmark40)



Рис. 4.30: Проверка местонахождения файлов

Копирую первую лабораторную с помощью утилиты cp и проверяю правильность выполнения команды cp с помощью ls (рис. [4.31).](#_bookmark41)

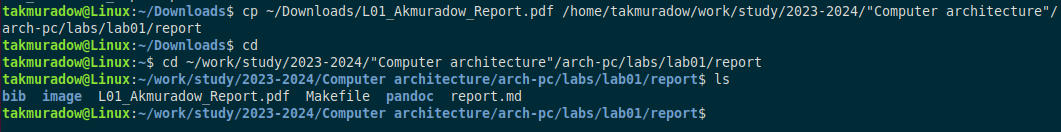


Рис. 4.31: Копирование файла

Добавляю файл L01\_Akmuradow\_Report.pdf (рис. [4.32).](#_bookmark46)



Рис. 4.32: Добавление файла на сервер

Сохраняю изменения на сервере командой git commit -m “…”, поясняя, что добавил файлы.

То же самое делаю для отчета по третьей лабораторной работе: перехожу в директорию labs/lab02/report с помощью cd, добавляю с помощью git add нужный файл, сохраняю изменения с помощью git commit (рис. [4.33).](#_bookmark47)

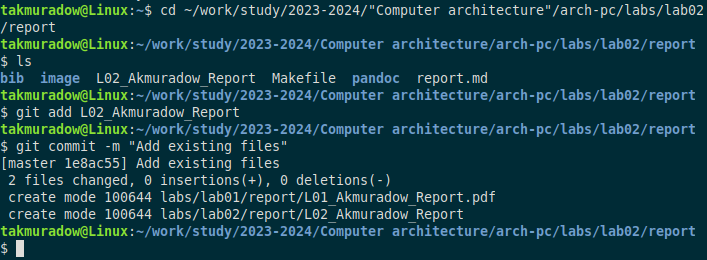


Рис. 4.33: Подкаталоги и файлы в репозитории

Отправляю в центральный репозиторий сохраненные изменения командой git push -f origin master (рис. [4.34).](#_bookmark48)

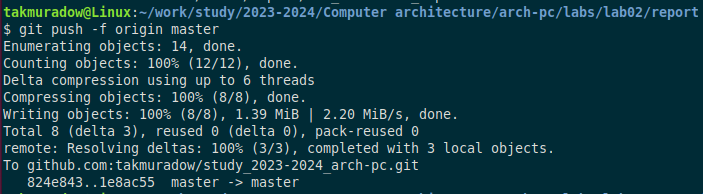


Рис. 4.34 Отправка в центральный репозиторий сохраненных изменений

Проверяю на сайте GitHub правильность выполнения заданий. Вижу, что пояснение к совершенным действиям отображается (рис. [4.35).](#_bookmark49)

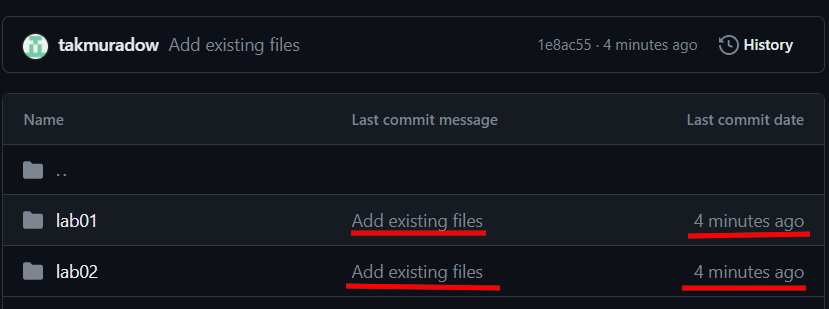


Рис. 4.35: Страница каталога в репозитории

При просмотре изменений так же вижу, что были добавлены файлы с отчетами по лабораторным работам (рис. [4.36).](#_bookmark50)

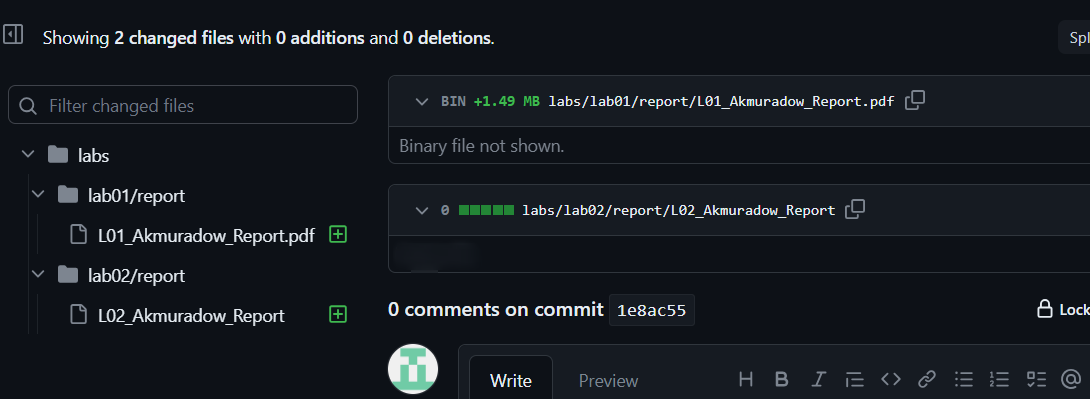


Рис. 4.36: Страница последних изменений в репозитории

Вижу, что отчеты по лабораторным работам находятся в соответствующих каталогах репозитория

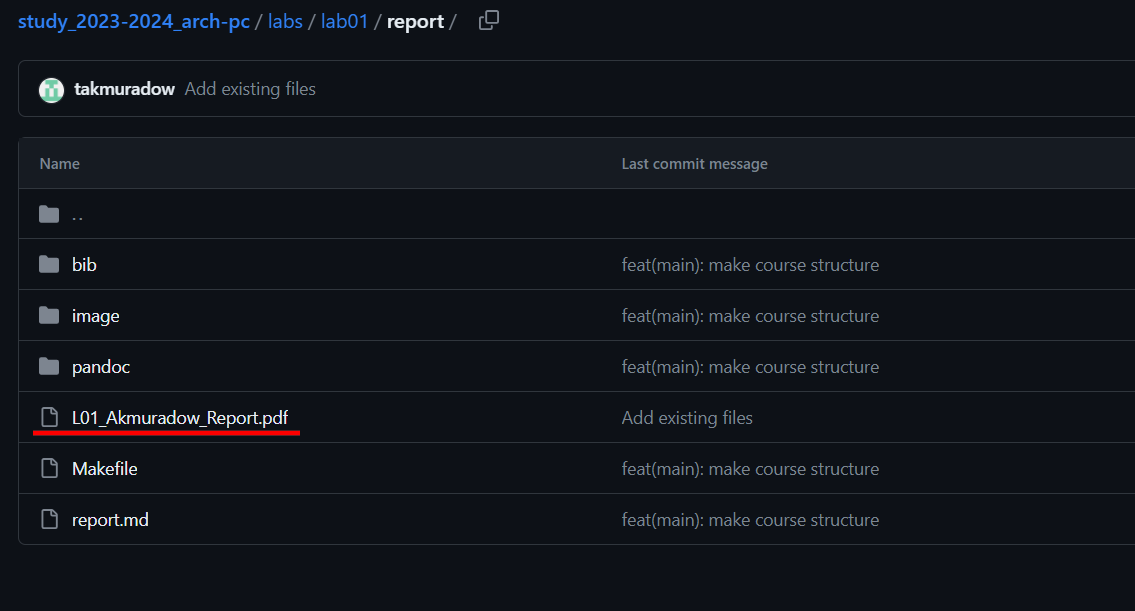


Рис. 4.37: Каталог lab01/report

# Выводы

При выполнении данной лабораторной работы я изучил идеологию и применение средств контроля версий, а также приобрел практические навыки по работе с системой git.

# Список литературы

* 1. [Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/1584622/mod_resource/content/1/%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%963.pdf)
  2. [Git - gitattributes Документация](https://git-scm.com/docs/gitattributes)