**Отчёт по лабораторной работе №2**

#### Дисциплина: архитектура компьютера

### Корчагин Алексей Павлович нмм-бд 02-23

# 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 работы.

# Теоретическое введение

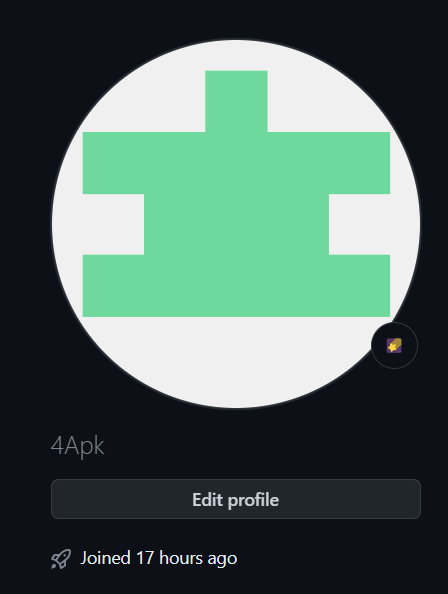
Системы контроля версий (Version Control System, VCS) применяются при ра- боте нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен до- ступ для участников проекта. При внесении изменений в содержание проекта система 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 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 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

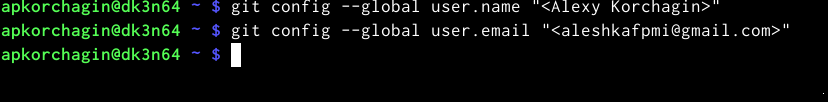
## Создание аккаунта GitHub

Создал аккаунт

Рис. 4.1:

## Базовая настройка Git

Открываю виртуальную машину, затем открываю терминал и делаю пред- варительную 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. 4.2: Предварительная конфигурация git

Настраиваю utf-8 в выводе сообщений git для корректного отображения сим- волов (рис. [4.2).](#_bookmark9)



Рис. 4.3: Настройка кодировки Задаю имя «master» для начальной ветки (рис. [4.4).](#_bookmark10)

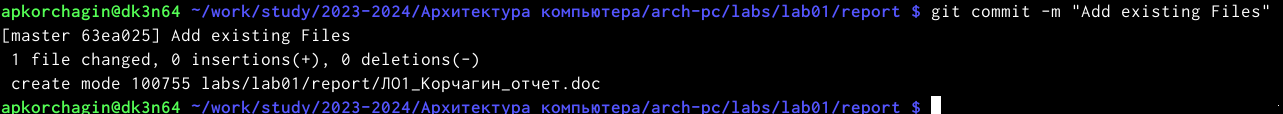


Рис. 4.4: Создание имени для начальной ветки

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 сим- волы, которые можно использовать для обозначения разрыва строки в текстовых файлах.



Рис. 4.5: Параметр autocrlf

Задаю параметр safecrlf со значением warn, так Git будет проверять преоб- разование на обратимость (рис. [4.6).](#_bookmark12) При значении warn Git только выведет предупреждение, но будет принимать необратимые конвертации.



Рис. 4.6: Параметр safecrlf

## Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозиториев необ- ходимо 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. [4.7).](#_bookmark14) Ключ автоматически сохранится в каталоге ~/.ssh/.

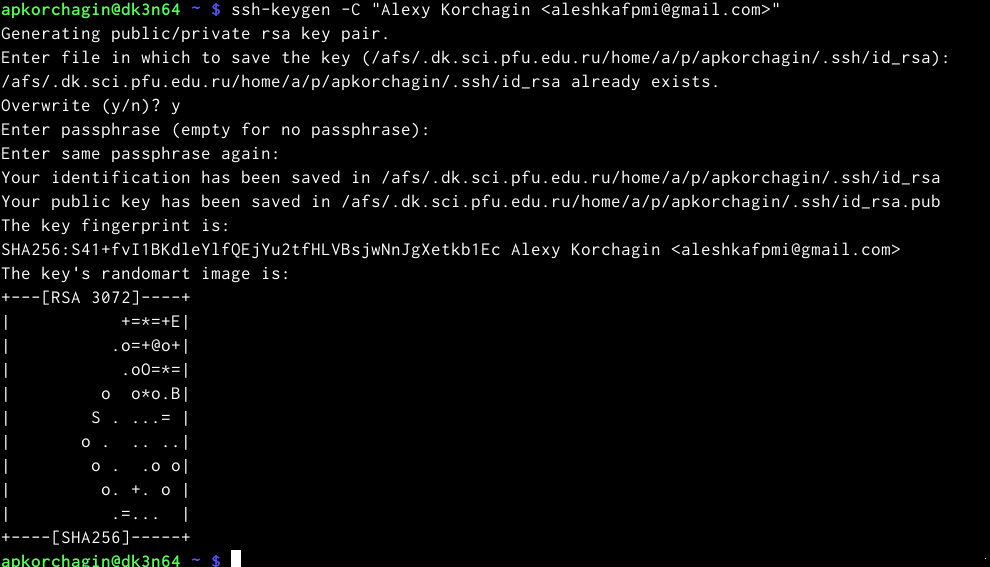


Рис. 4.7: Генерация SSH-ключа



Рис. 4.8: Копирование ключа из консоли

Открываю браузер, захожу на сайт GitHub. Открываю свой профиль и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key» (рис. [4.9).](#_bookmark17)

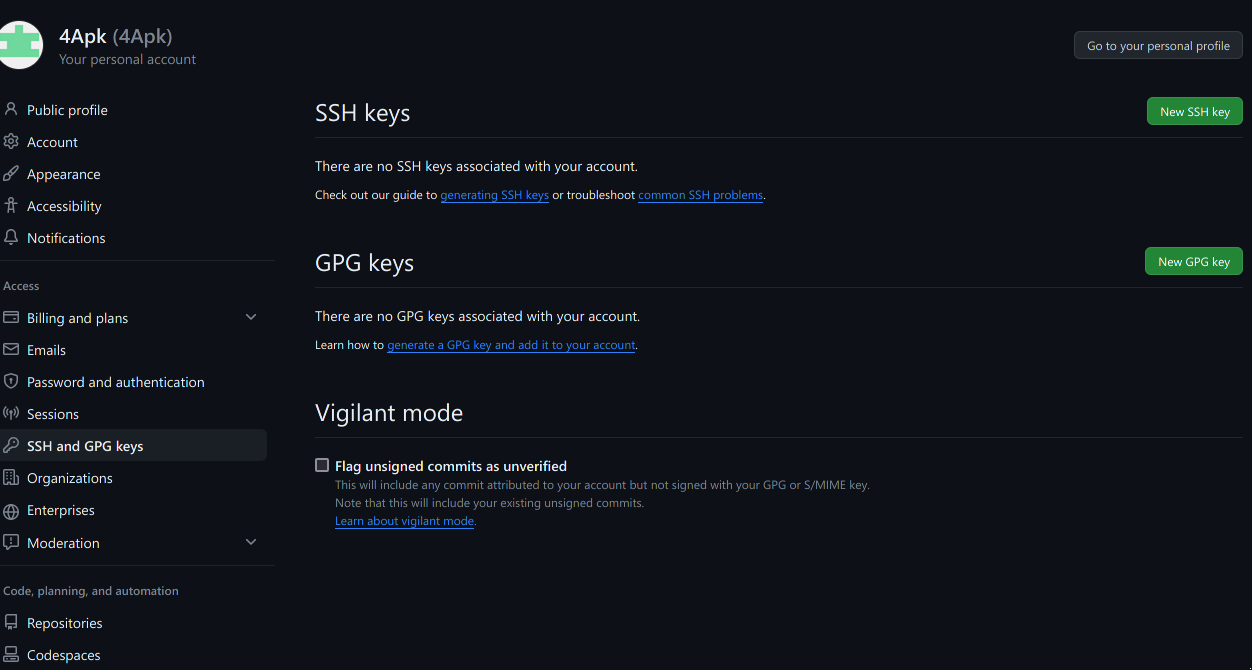


Рис. 4.9: Окно SSH and GPG keys

Вставляю скопированный ключ в поле «Key». В поле Title указываю имя для ключа. Нажимаю «Add SSH-key», чтобы завершить добавление ключа (рис. [4.10).](#_bookmark18)

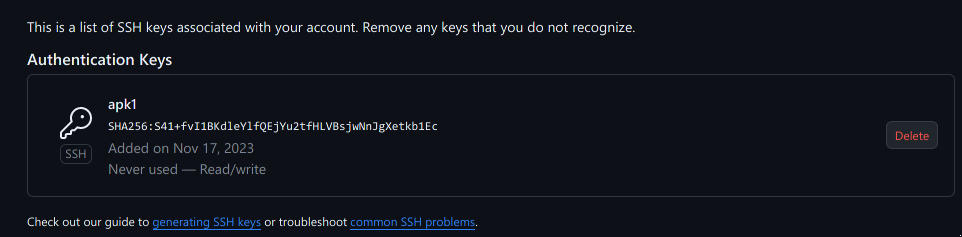


Рис. 4.10: Добавил ключ

## Создание рабочего пространства и репозитория курса на основе шаблона

Закрываю браузер, открываю терминал. Создаю директорию, рабочее про- странство, с помощью утилиты mkdir, блягодаря ключу -p создаю все директории после домашней ~/work/study/2023-2024/“Архитектура компьютера” рекурсивно. Далее проверяю с помощью ls, действительно ли были созданы необходимые мне каталоги (рис. [4.11).](#_bookmark20)



Рис. 4.11: 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. [4.12).](#_bookmark22)

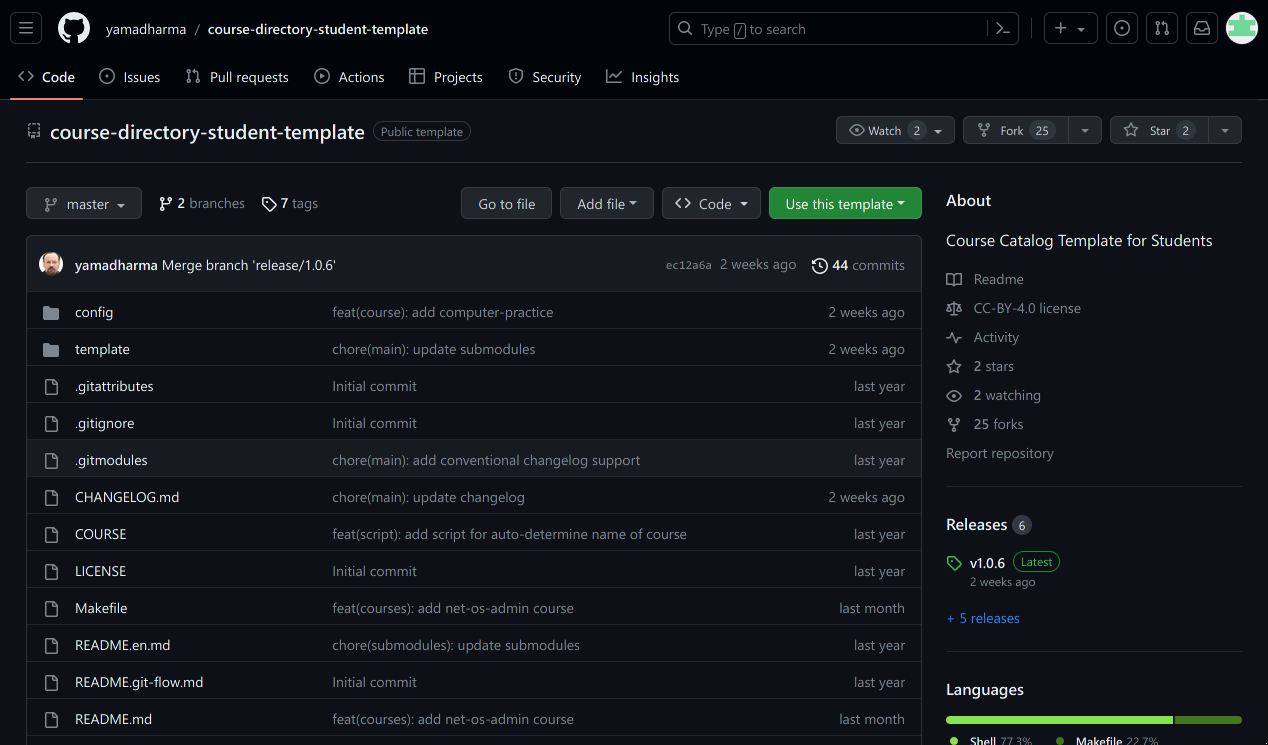
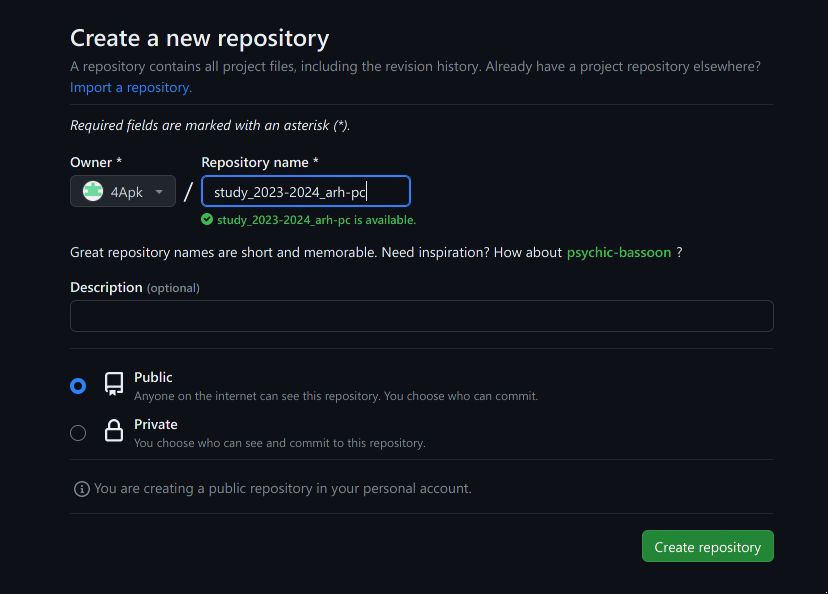


Рис. 4.12: Страница шаблона для репозитория

В открывшемся окне задаю имя репозитория (Repository name): study\_2023–2024\_arh- pc и создаю репозиторий, нажимаю на кнопку «Create repository from template»

(рис. [4.13).](#_bookmark23)



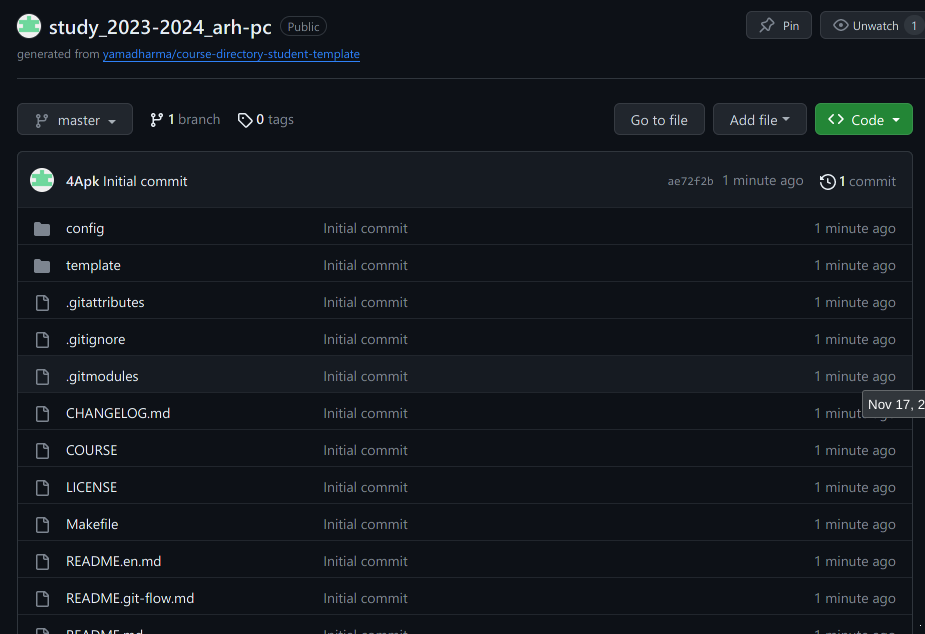
Рис. 4.13: Окно создания репозитория Репозиторий создан (рис. [4.14).](#_bookmark24)

Рис. 4.14: Созданный репозиторий

Через терминал перехожу в созданный каталог курса с помощью утилиты cd (рис. [4.15).](#_bookmark25)



Рис. 4.15: Перемещение между директориями

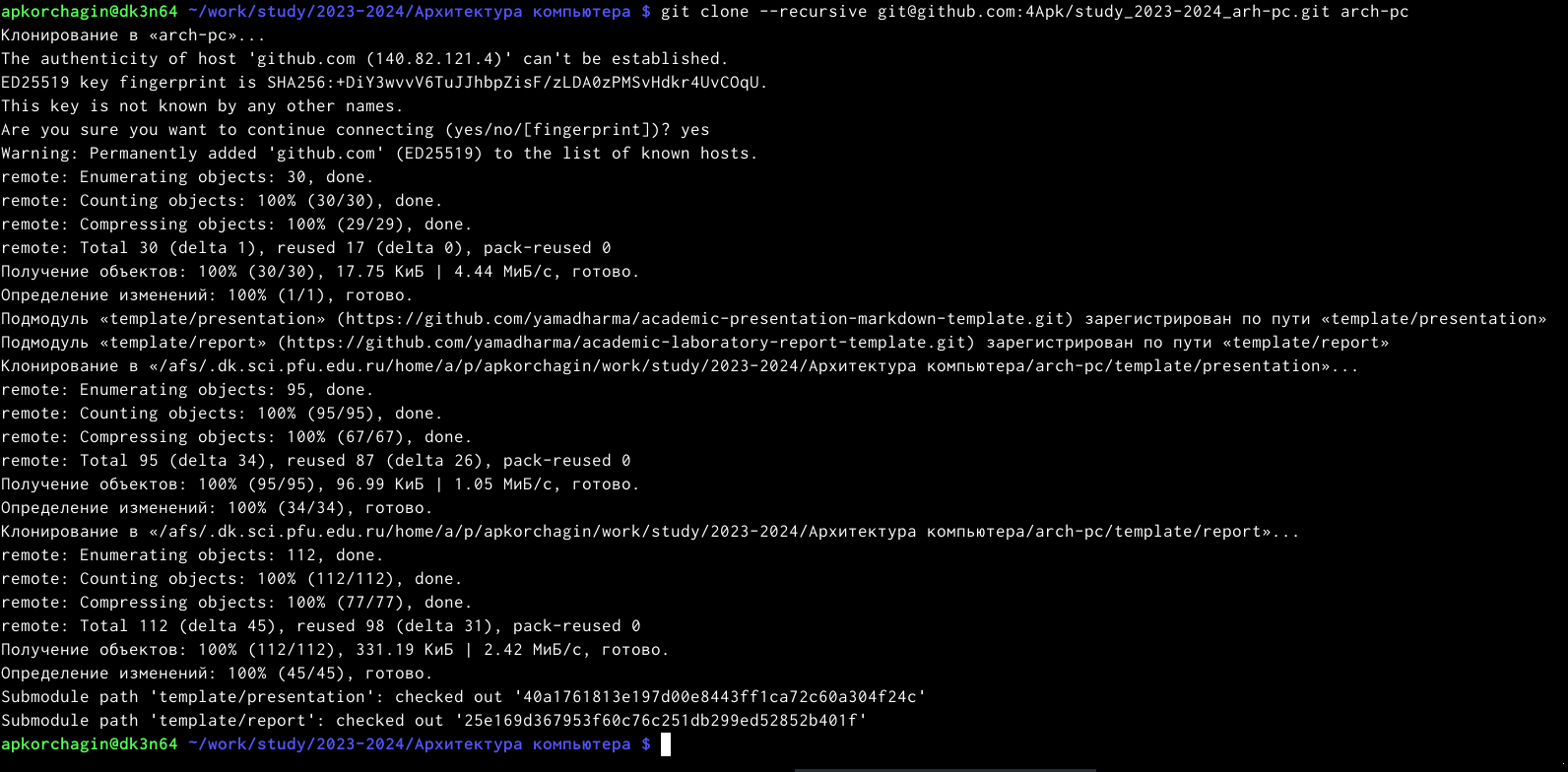
Клонирую созданный репозиторий с помощью команды git clone –recursive git@github.com:/study\_2023–2024\_arh-pc.git arch-pc (рис. [4.16).](#_bookmark26)

Рис. 4.16: Клонирование репозитория

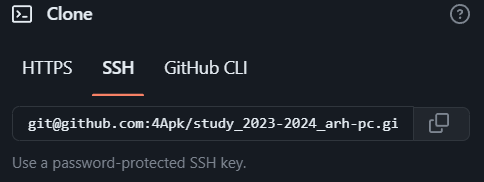
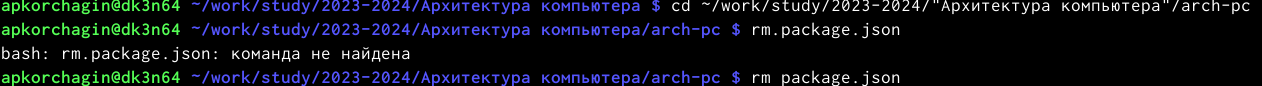
Копирую ссылку для клонирования на странице созданного репозитория, сначала перейдя в окно «code», далее выбрав в окне вкладку “SSH” (рис 4.17)

Рис. 4.17:Ссылка на клонирование

## Настройка каталога курса

Перехожу в каталог arch-pc с помощью утилиты cd и удаляю лишние файлы при помощи rm (рис. [4.18) .](#_bookmark29)



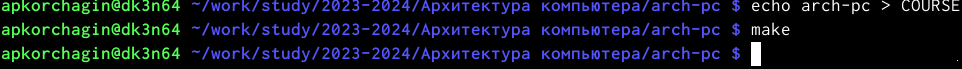
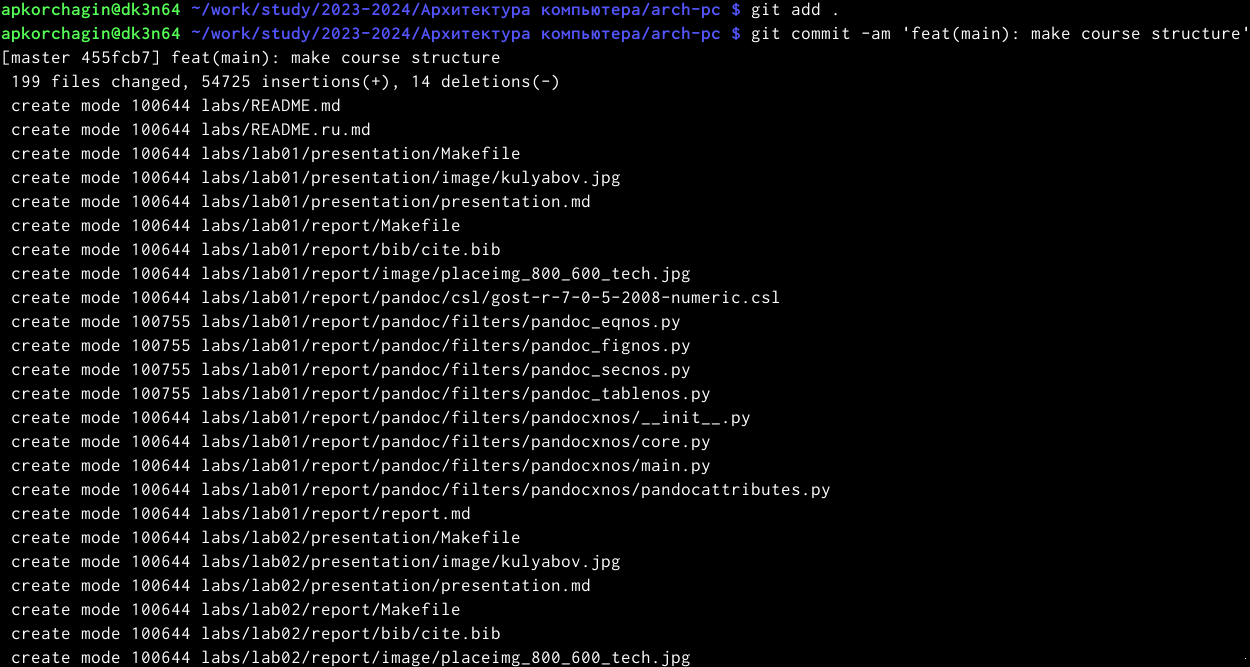
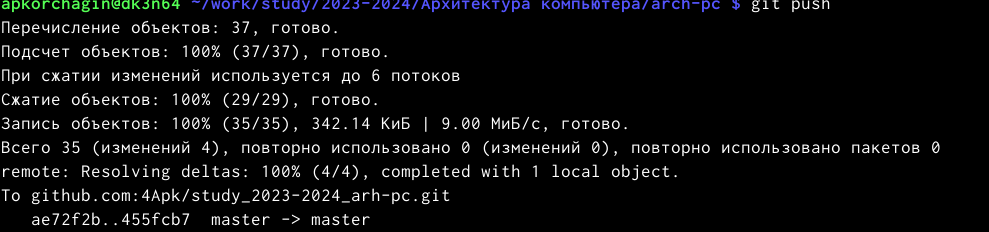
Рис. 4.18: Удаление файлов Создаю необходимые каталоги (рис. [4.19).](#_bookmark31)

Рис. 4.19: 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. [4.20).](#_bookmark32)

Отправляю все на сервер с помощью push (рис. [4.21).](#_bookmark33)



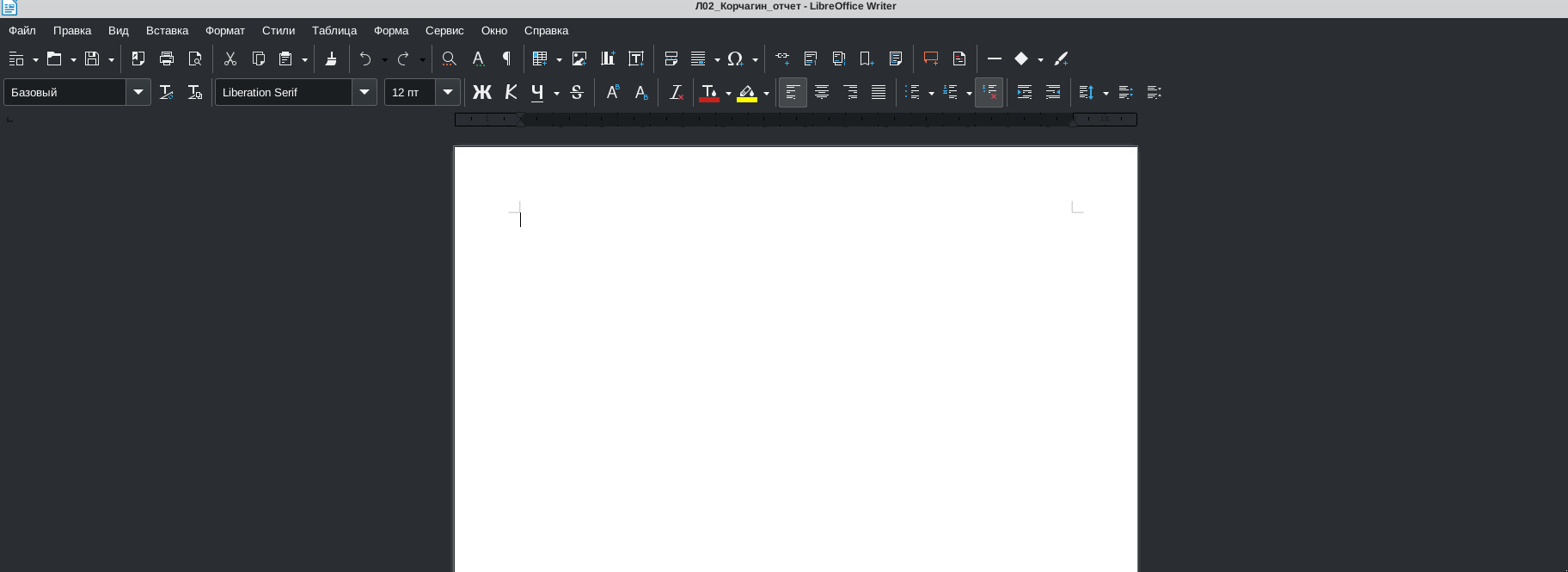
Проверяю правильность выполнения работы сначала на самом сайте GitHub (рис. [4.22).](#_bookmark34)

## Выполнение заданий для самостоятельной работы

* + 1. Перехожу в директорию labs/lab02/report с помощью утилиты cd. Создаю в каталоге файл для отчета по второй лабораторной работе с помощью утилиты touch (рис. [4.23).](#_bookmark36)



Рис. 4.23: Создание файла

Оформить отчет я смогу в текстовом процессоре LibreOffice Writer, найдя его в меню приложений (рис. [4.24).](#_bookmark37)

* + 1. Перехожу из подкаталога lab02/report в подкаталог lab01/report с помощью утилиты cd (рис. [4.25).](#_bookmark39)

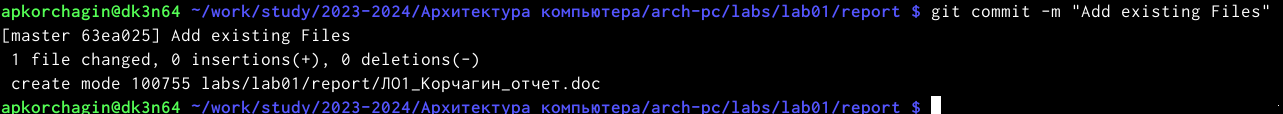
Проверяю местонахождение файла с отчетом по первой лабораторной работе. Он должен быть в подкаталоге домашней директории «Загрузки», для проверки использую команду ls (рис. [4.26).](#_bookmark40)

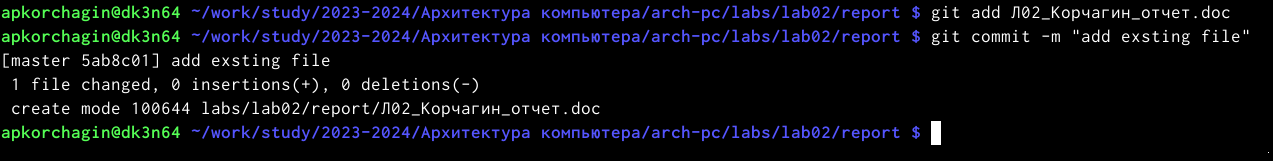


Копирую первую лабораторную с помощью утилиты cp и проверяю правиль- ность выполнения команды cp с помощью ls (рис. [4.27).](#_bookmark41)

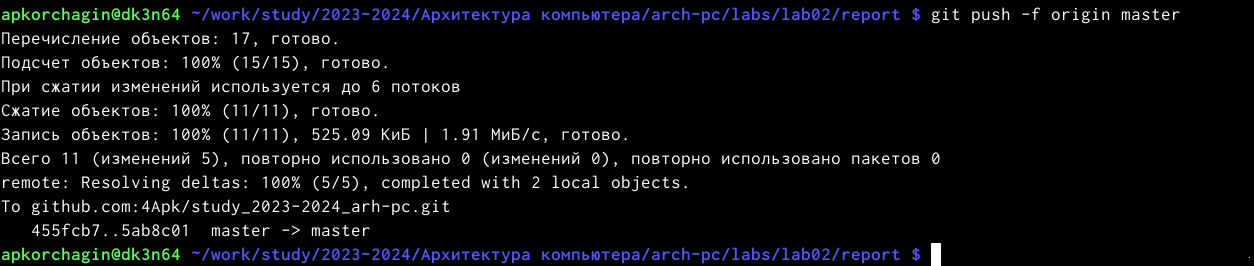


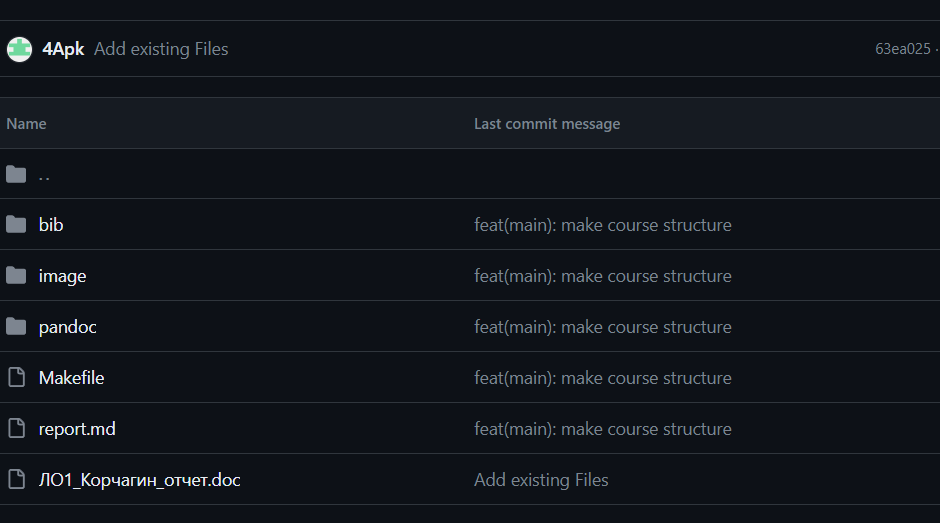
* + 1. Добавляю с помощью команды git add в коммит созданные файлы: Л01\_Корчагин\_отчет (рис. [4.28).](#_bookmark44)

Сохраняю изменения на сервере командой git commit -m “…”, поясняя, что добавила файлы(4.29).

То же самое делаю для отчета по второй 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. [4.30).](#_bookmark47)

Отправляю в центральный репозиторий сохраненные изменения командой git push -f origin master (рис. [4.31).](#_bookmark48)



При просмотре изменений на Github вижу, что были добавлены файлы с отчетами по лабораторным работам (рис. [4.32-4.33) .](#_bookmark50)

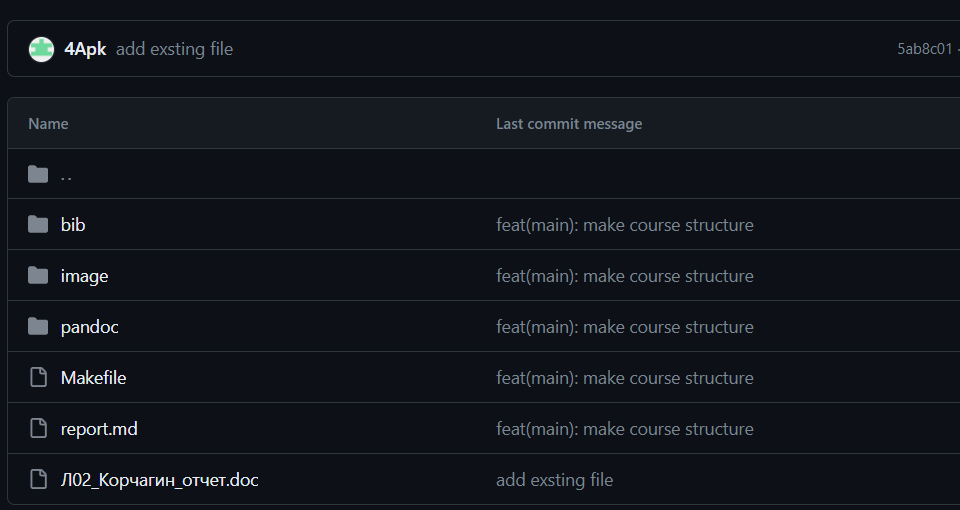


Рис. 4.33: Каталог lab02/report

# Выводы

При выполнении данной лабораторной работы я изучила идеологию и при- менение средств контроля версий, а также приобрела практические навыки по работе с системой git.