**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

Дисциплина: Архитектура компьютеров и операционные системы

Студент: Клименко Алёна

Группа: НКАбд-02-24

**МОСКВА**

2024 г.

**Оглавление**

[1 Цель работы 3](#_Toc178432138)

[2 Задание 4](#_Toc178432139)

[3 Теоретическое введение 5](#_Toc178432140)

[4 Выполнение лабораторной работы 6](#_Toc178432141)

[4.1 Настройка GitHub 6](#_Toc178432142)

[4.2 Базовая настройка git. 6](#_Toc178432143)

[4.3. Создание SSH ключа 7](#_Toc178432144)

[4.4. Создание рабочего пространства, репозитория курса на основе шаблона 8](#_Toc178432145)

[4.5. Создание репозитория курса на основе шаблона. 9](#_Toc178432146)

[4.6. Настройка каталога курса. 10](#_Toc178432147)

[5. Выполнение заданий для самостоятельной работы. 11](#_Toc178432148)

[5. Заключение 14](#_Toc178432149)

[6. Список используемой литературы 15](#_Toc178432150)

# 1 Цель работы

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

# 2 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

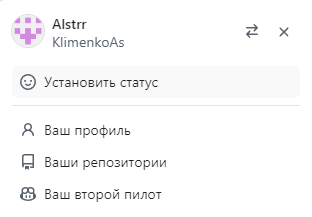
# 3 Теоретическое введение

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. 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.

# 4 Выполнение лабораторной работы

## 4.1 Настройка GitHub

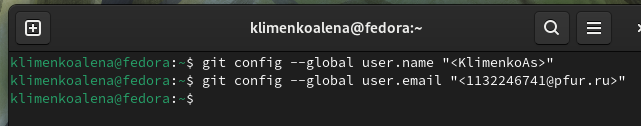
Создала учетную запись на сайте GitHub и заполнила основные данные учетной записи. (рис. 4.1.1)



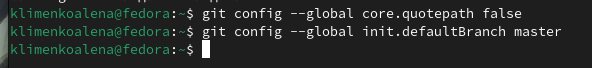
(рис. 4.1.1)

## 4.2 Базовая настройка git.

Открываю виртуальную машину, затем открываю терминал и делаю предварительную конфигурацию git. Ввожу команду git config –global user.name “”, указывая свое имя и команду git config –global user.email “[work@mail](mailto:work@mail)”, указывая в ней электронную почту (рис.4.2.1)



(рис.4.2.1)

Настраиваю utf-8 в выходе сообщений git. (рис.4.2.2) Задаю имя начальной ветки, которую буду называть master (рис. 4.2.2) 

(рис.4.2.2)

А также ввожу autocrlf (рис. 4.2.3)



(рис. 4.2.3)

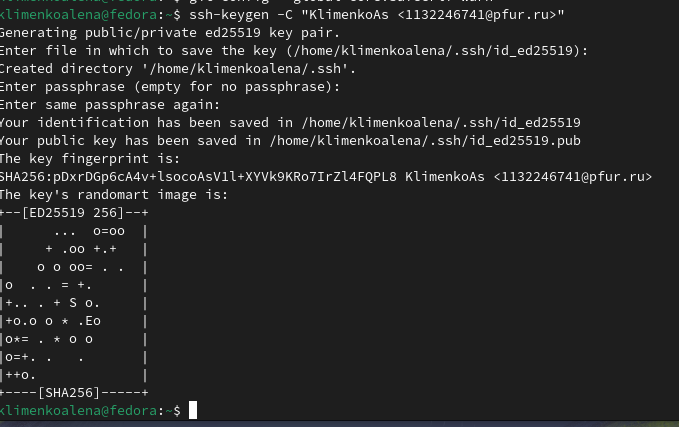
ввожу safecrlf (рис. 4.2.4)



(рис. 4.2.4)

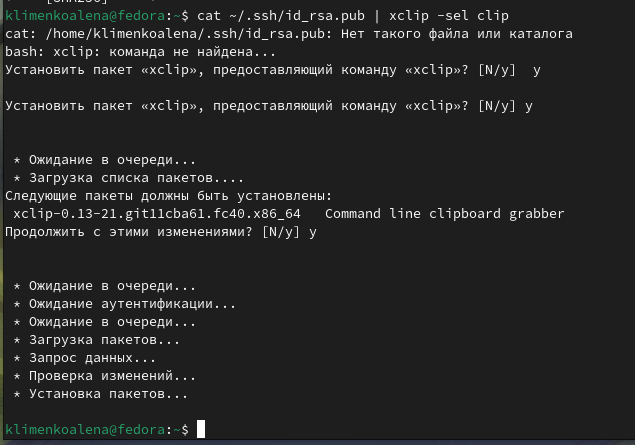
## 4.3. Создание SSH ключа

Для последующей идентификации пользователя на сервере репозиториев генерирую пару ключей (приватный и открытый). (рис. 4.3.1)



(рис. 4.3.1)

Чтобы скопировать из локальной консоли ключ в буфер обмена, устанавливаю команду xclip (рис. 4.3.2).



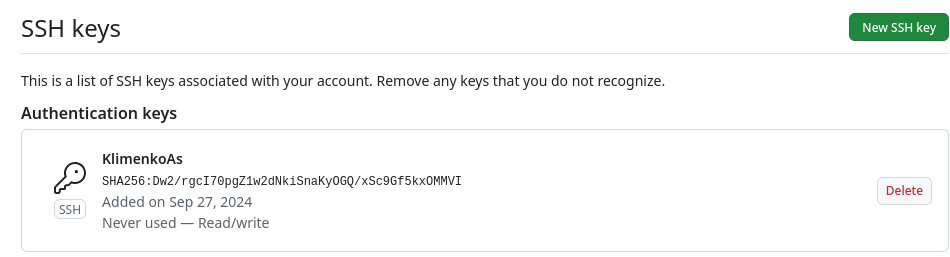
(рис. 4.3.2)

Теперь воспользуюсь командой xclip (рис. 4.3.3).



(рис. 4.3.3)

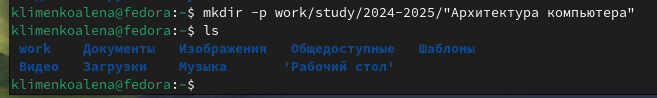
Вставляю ключ в появившееся на сайте поле и указываю его имя. (рис. 4.3.4)



(рис. 4.3.4)

## 4.4. Создание рабочего пространства, репозитория курса на основе шаблона

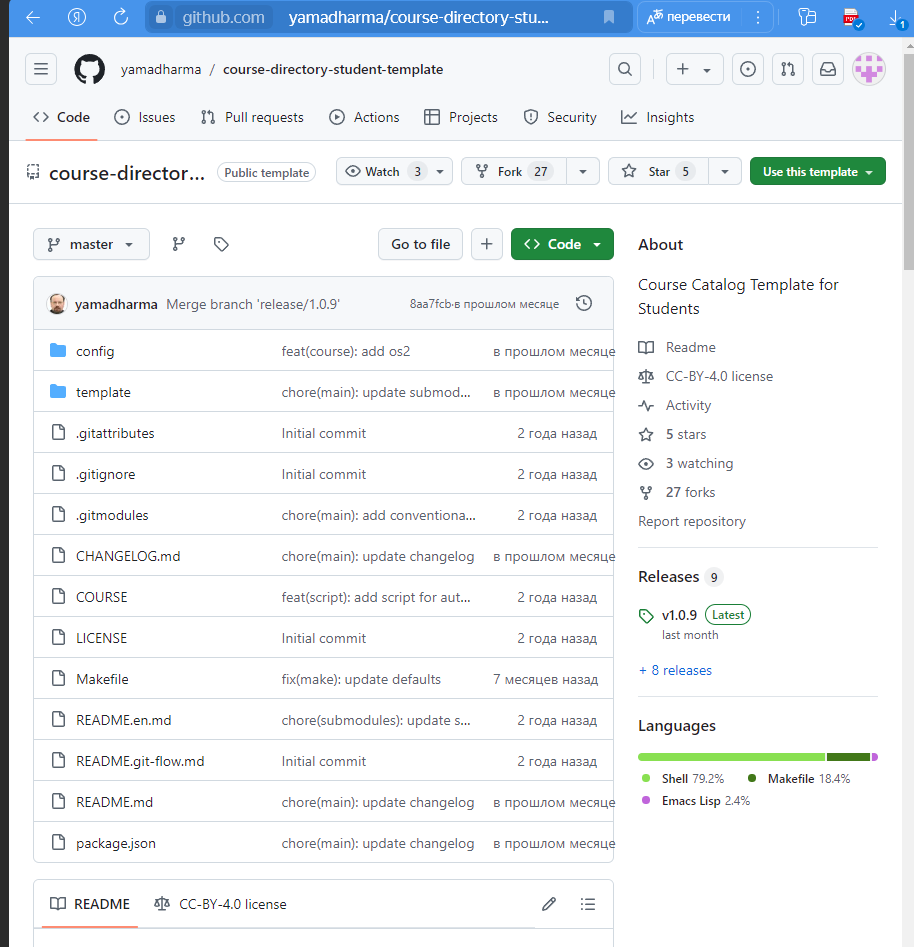
Открываю терминал и создаю репозиторий для предмета «Архитектура компьютера». (рис. 4.4.1.)



(рис. 4.4.1.)

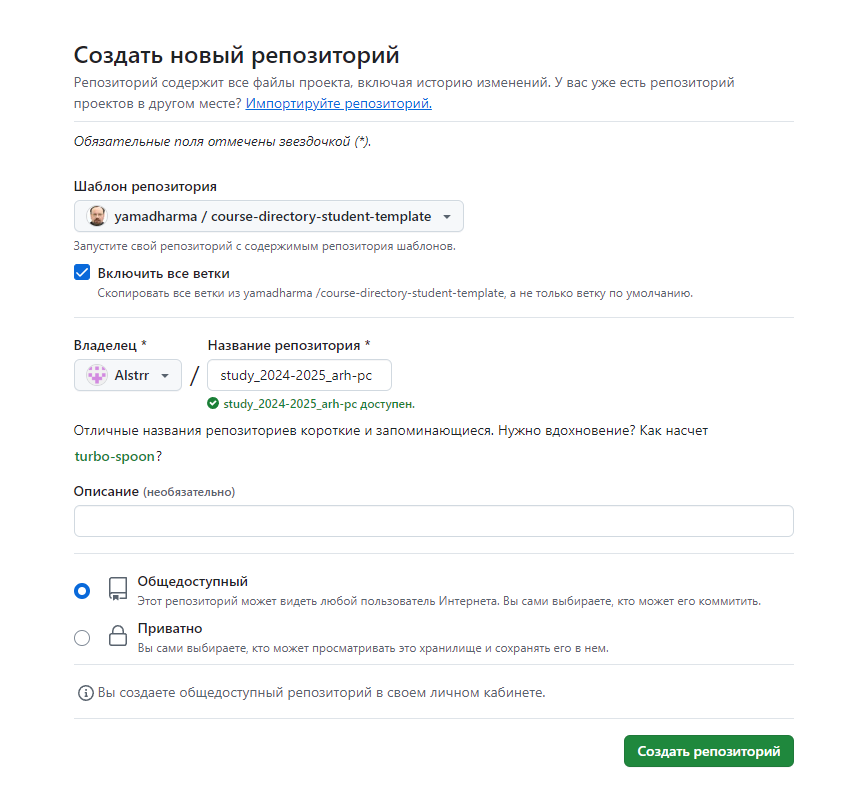
## 4.5. Создание репозитория курса на основе шаблона.

Захожу на страницу репозитория с шаблоном курса, выбираю его в качестве своего нового. (рис. 4.5.1.)



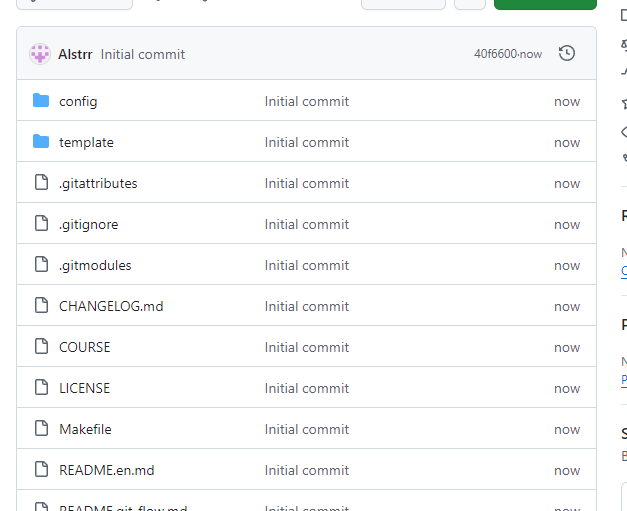
.(рис. 4.5.1.)

Далее создаю его и задаю ему имя. (рис. 4.5.2.)



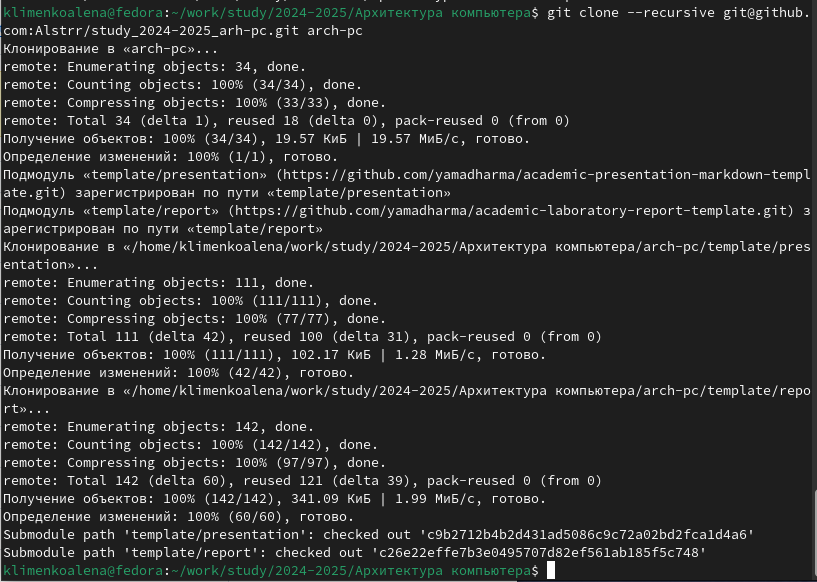
(рис. 4.5.2.)

Проверяю корректность создания (рис. 4.5.3.)



(рис. 4.5.3.)

Клонирую созданный репозиторий (рис. 4.5.4.)



(рис. 4.5.4.)

## 4.6. Настройка каталога курса.

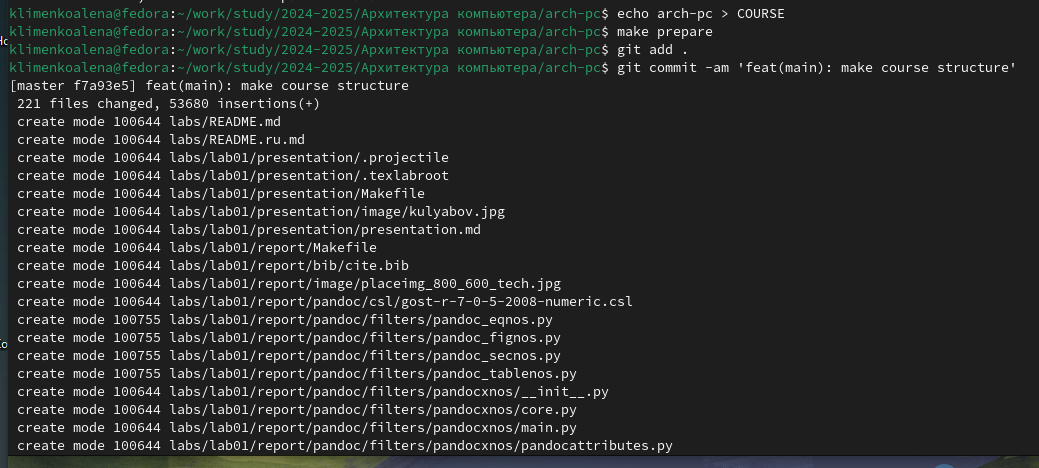
Перехожу в каталог курса(рис. 4.6.1.) и удаляю лишний файлы. (рис. 4.6.2.)

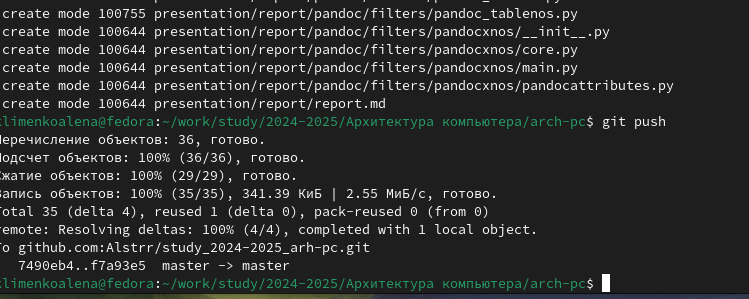


(рис. 4.6.1.)

(рис. 4.6.2.)

Создаю необходимые каталоги, отправляю файлы на сервер. (рис. 4.6.3.) , (рис. 4.6.4.),

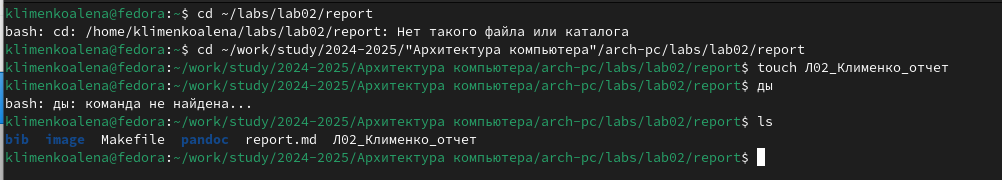


(рис. 4.6.3.) 

(рис. 4.6.4.)

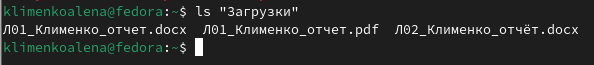
## 5. Выполнение заданий для самостоятельной работы.

Создаю отчет по выполнению второй лабораторной работы в соответствующем каталоге. С помощью команды ls проверяю, создан ли файл. (рис. 4.5.1)



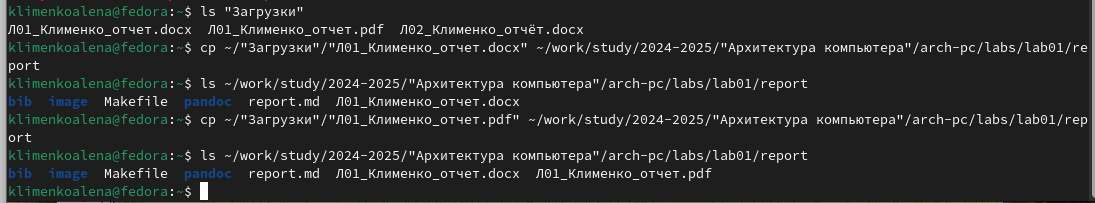
(рис. 4.5.1)

проверяю местонахождение своих лабораторных работ. (рис. 4.5.2)



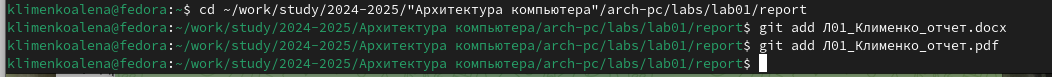
(рис. 4.5.2)

Копирую лабораторную работу с помощью утилиты cp, проверяю местонахождение файлов с помощью команды ls. (рис.4.5.3)



(рис.4.5.3)

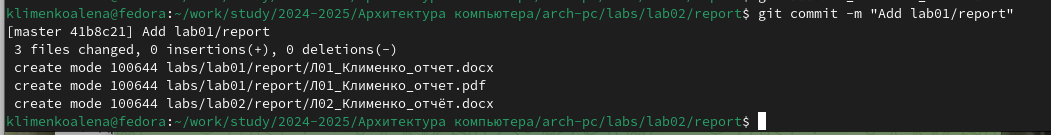
Для того чтобы загрузить эти файлы на GitHub, в первую очередь я использую команду git add. Так добавленные мной файлы станут отслеживаемыми. (рис.4.5.4)



(рис.4.5.4)

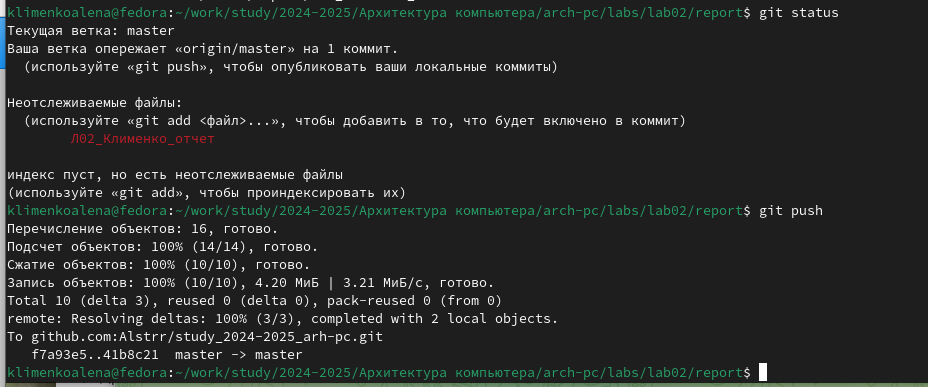


Теперь осуществляю полноценный перенос файлов (рис. 4.5.5.)



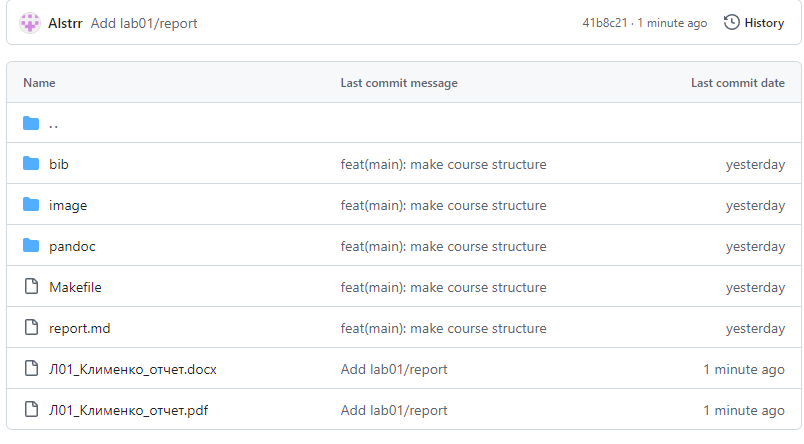
(рис. 4.5.5.)

Использую команды: git status и git push, чтобы опубликовать свои локальные коммиты. (рис. 4.5.6.)



(рис. 4.5.6.)

Перехожу в каталоги на GitHub, чтобы убедиться в том, что файлы находятся в нужных репозиториях. (рис. 4.5.7.)



(рис. 4.5.7.)

# 5. Заключение

В заключение хочется отметить, что данная лабораторная работа позволила мне научиться работать с системой Git. Я практиковала свои навыки в работе с командной строкой, теперь уже связывая выполнимое с директориями GitHub.

# 6. Список используемой литературы

1. Туис:

- **Курс: Архитектура компьютеров и операционные системы.**

<https://esystem.rudn.ru/course/view.php?id=112>

- **Пример оформления отчёта по лабораторной работе** <https://esystem.rudn.ru/pluginfile.php/2088992/mod_resource/content/0/Пример%20оформления%20отчета%20по%20лабораторной%20работе.pdf>

- **Лабораторная работа №2. Основы интерфейса командной строки ОС GNU Linux** <https://esystem.rudn.ru/pluginfile.php/2089080/mod_resource/content/0/Лабораторная%20работа%20№2.%20Основы%20интерфейса%20командной%20строки%20ОС%20GNU%20Linux.pdf>

2. GitHab:

https://github.com/