Отчёт по лабораторной работе №2

Дисциплина: архитектура компьютеров и операционные системы

Постнова Елизавета Андреевна

Содержание

# 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 github.
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. Задания 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. 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. 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.

В табл. [1](#tbl:std-dir) приведены основные команды git.

Table 1: Основные команды git

| Команда | Описание |
| --- | --- |
| git init | создание основного дерева репозитория |
| git pull | получение обновлений (изменений) текущего дерева из центрального |
| репозитория |  |
| git push | отправка всех произведённых изменений локального дерева в |
| центральный репозиторий |  |
| git status | просмотр списка изменённых файлов в текущей директории |
| git diff | просмотр текущих изменений |
| git add . | добавить все изменённые и/или созданные файлы и/или каталоги |
| git add имена\_файлов | добавить конкретные изменённые и/или созданные файлы и/или |
| каталоги |  |
| git rm имена\_файлов | удалить файл и/или каталог из индекса репозитория (при этом файл |
| и/или каталог остаётся в локальной директории) |  |
| git commit -am 'Описание коммита' | сохранить все добавленные изменения и все изменённые файлы |
| git checkout -b имя\_ветки | создание новой ветки, базирующейся на текущей |
| git checkout имя\_ветки | переключение на некоторую ветку (при переключении на ветку, |

которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) | | git push origin имя\_ветки | потправка изменений конкретной ветки в центральный репозиторий | | git merge --no-ff имя\_ветки | слияние ветки с текущим деревом | | git branch -d имя\_ветки | удаление локальной уже слитой с основным деревом ветки | | git branch -D имя\_ветки | принудительное удаление локальной ветки | | git push origin :имя\_ветки | удаление ветки с центрального репозитория |

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): git checkout master git pull git checkout -b имя\_ветки 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. Для этого необходимо проверить, какие файлы изменились к текущему моменту: git status и при необходимости удаляем лишние файлы, которые не хотим отправлять в центральный репозиторий. Затем полезно просмотреть текст изменений на предмет соответствия правилам ведения чистых коммитов: git diff Если какие-либо файлы не должны попасть в коммит, то помечаем только те файлы, изменения которых нужно сохранить. Для этого используем команды добавления и/или удаления с нужными опциями: git add имена\_файлов git rm имена\_файлов Если нужно сохранить все изменения в текущем каталоге, то используем: git add Затем сохраняем изменения, поясняя, что было сделано: git commit -am “Some commit message” и отправляем в центральный репозиторий: git push origin имя\_ветки или git push

# 4 Выполнение лабораторной работы

## 4.1 **Настройка github**

Создаю учётную запись на сайте github и заполняю основные данные. (рис. [24](#fig:001)).

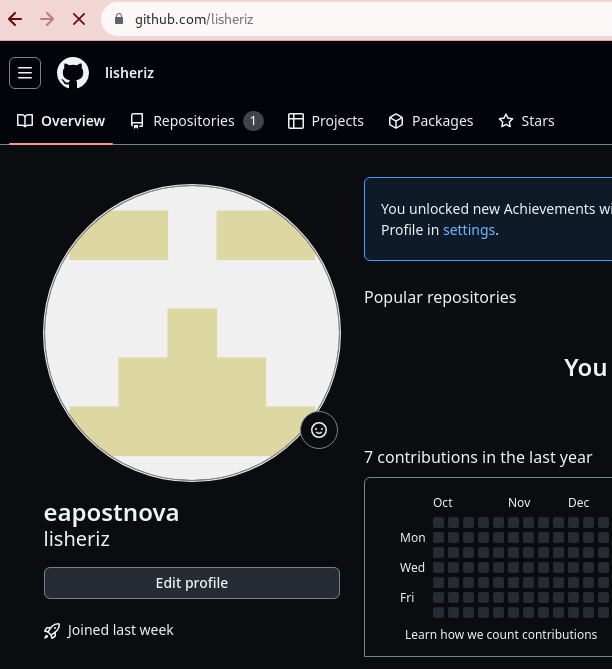


Figure 1: Создание учетной записи на github

## 4.2 **Базовая настройка git**

Делаю предварительную конфигурацию git. Открываю терминал и ввожу следующие команды, указав свои имя и email на github. (рис. [24](#fig:001)).

Figure 2: Предварительная конфигурация git

Figure 2: Предварительная конфигурация git

Настраиваю utf-8 в выводе сообщений git. (рис. [24](#fig:001)).

Figure 3: Настройка utf-8

Figure 3: Настройка utf-8

Задаю имя начальной ветки. (рис. [24](#fig:001)).

Figure 4: Задание имени начальной ветки

Figure 4: Задание имени начальной ветки

Задаю параметры autocrlf и safecrlf. (рис. [24](#fig:001)).

Figure 5: Задание параметров autocrlf и safecrlf

Figure 5: Задание параметров autocrlf и safecrlf

## 4.3 **Создание SSH ключа**

Для последующей идентификации пользователя на сервере репозиториев сгенерирую пару ключей (приватный и открытый). (рис. [24](#fig:001)).

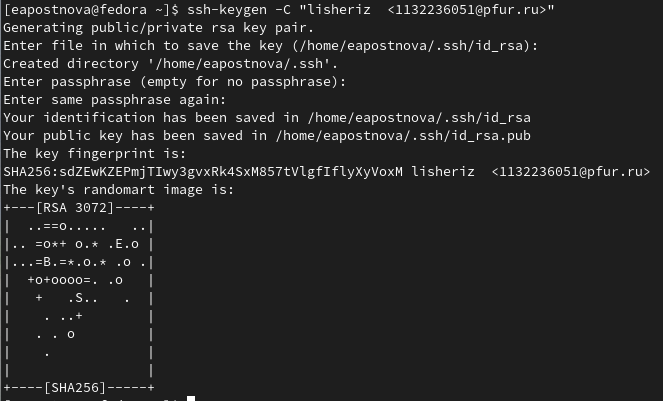


Figure 6: Генерация ключей

Далее загружаю сгенерированный открытый ключ. Для этого захожу на сайт github под своей учётной записью и перехожу в меню Setting. После этого выбираю в боковом меню SSH and GPG keys и нажимаю кнопку New SSH key. Скопировав из локальной консоли ключ в буфер обмена с помощью команды, (рис. [24](#fig:001)).

Figure 7: Получение ключа из локальной консоли

Figure 7: Получение ключа из локальной консоли

вставляю ключ в появившееся на сайте поле и указываю для ключа имя. (рис. [24](#fig:001)).

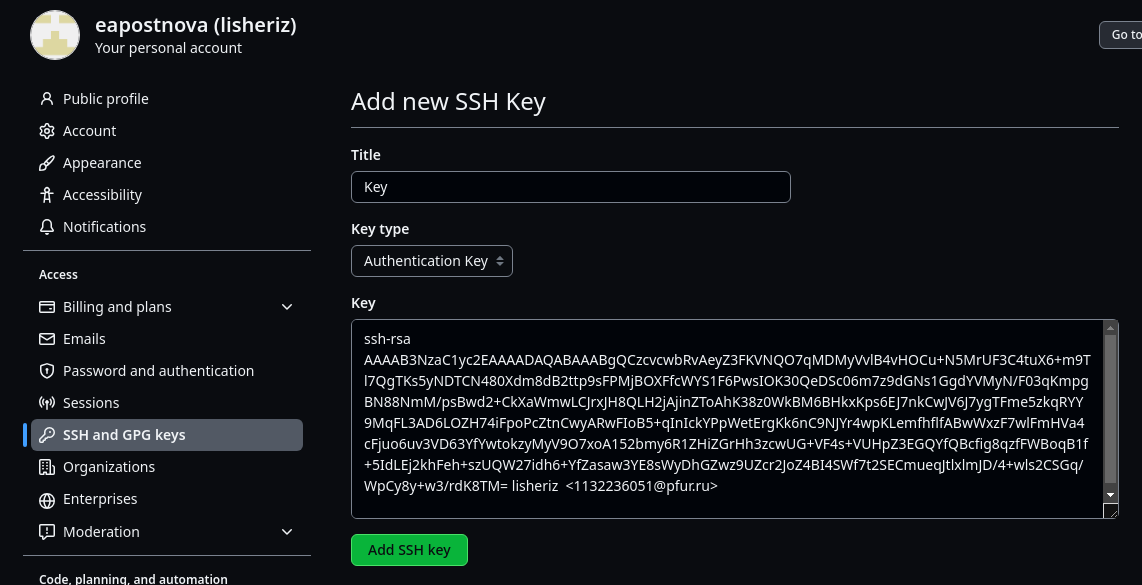


Figure 8: Загрузка ключа в github

## 4.4 **Создание рабочего пространства и репозитория курса на основе шаблона**

Открываю терминал и создаю каталог для предмета «Архитектура компьютера». (рис. [24](#fig:001)).

Figure 9: Создание каталога

Figure 9: Создание каталога

(рис. [24](#fig:001)).

Figure 10: Проверка выполнения команды

Figure 10: Проверка выполнения команды

## 4.5 **Создание репозитория курса на основе шаблона**

Перехожу на станицу репозитория с шаблоном курса и выбираю Use this template. В открывшемся окне задаю имя репозитория - study\_2023– 2024\_arhpc и создаю репозиторий. (рис. [24](#fig:001)).

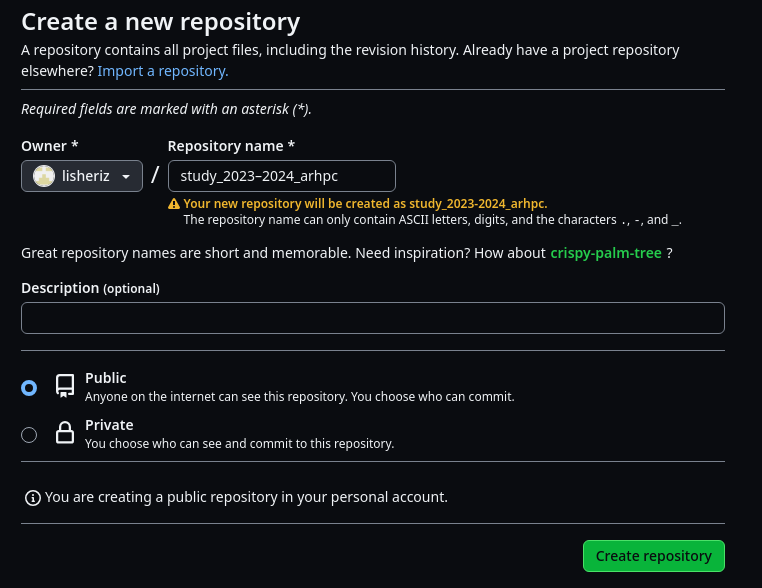


Figure 11: Создание репозитория

Открываю терминал, перехожу в каталог курса и клонирую созданный репозиторий. (рис. [24](#fig:001)).

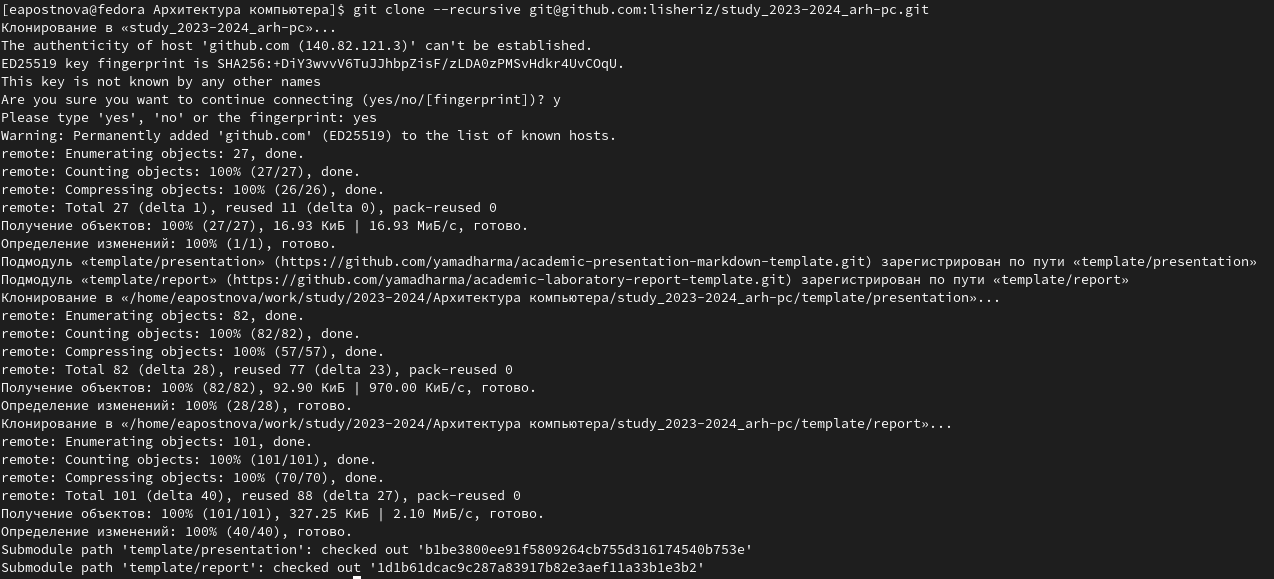


Figure 12: Клонирование репозитория

## 4.6 **Настройка каталога курса**

Перехожу в каталог курса, удаляю лишние файлы и создаю необходимые каталоги. (рис. [24](#fig:001)).

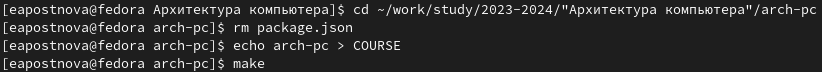


Figure 13: Удаление файлов и создание каталогов

Отправляю файлы на сервер.(рис. [24](#fig:001)).

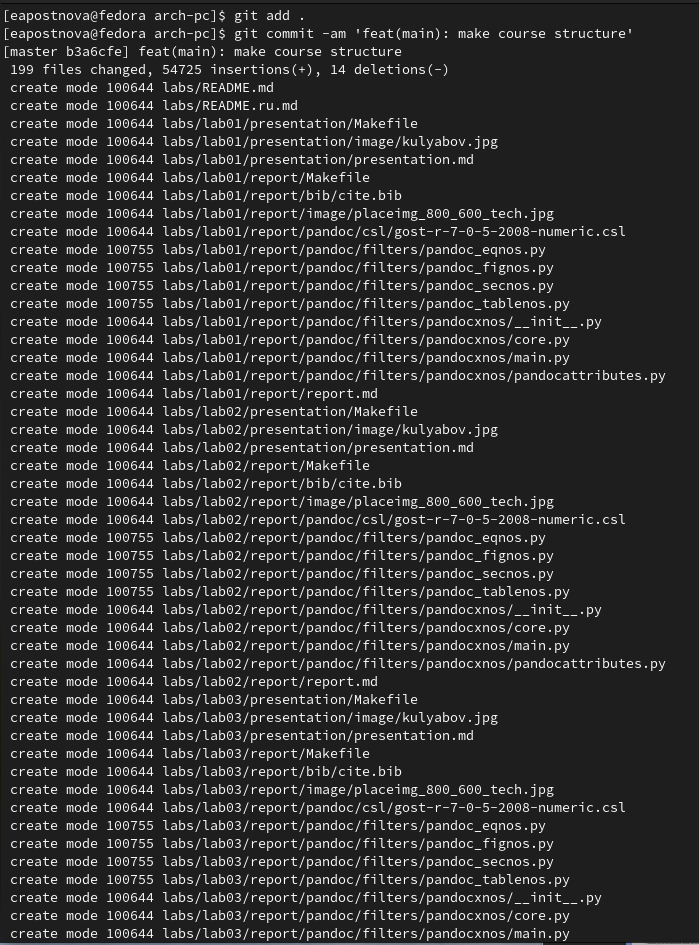


Figure 14: Ввод необходимых команд для отправки файлов на сервер

(рис. [24](#fig:001)).

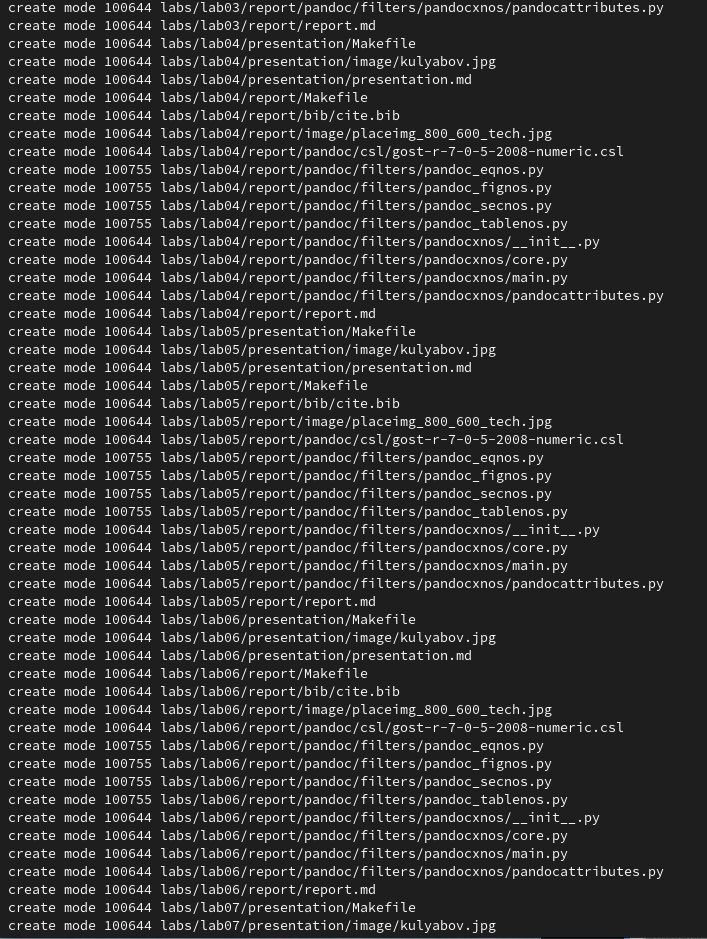


Figure 15: Ввод необходимых команд для отправки файлов на сервер

(рис. [24](#fig:001)).



Figure 16: Ввод необходимых команд для отправки файлов на сервер

(рис. [24](#fig:001)).



Figure 17: Ввод необходимых команд для отправки файлов на сервер

(рис. [24](#fig:001)).

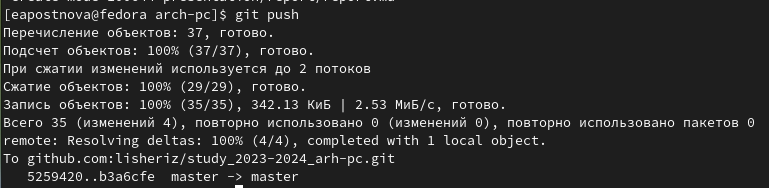


Figure 18: Ввод необходимых команд для отправки файлов на сервер

Проверяю правильность создания иерархии рабочего пространства в локальном репозитории и на странице github. (рис. [24](#fig:001)).

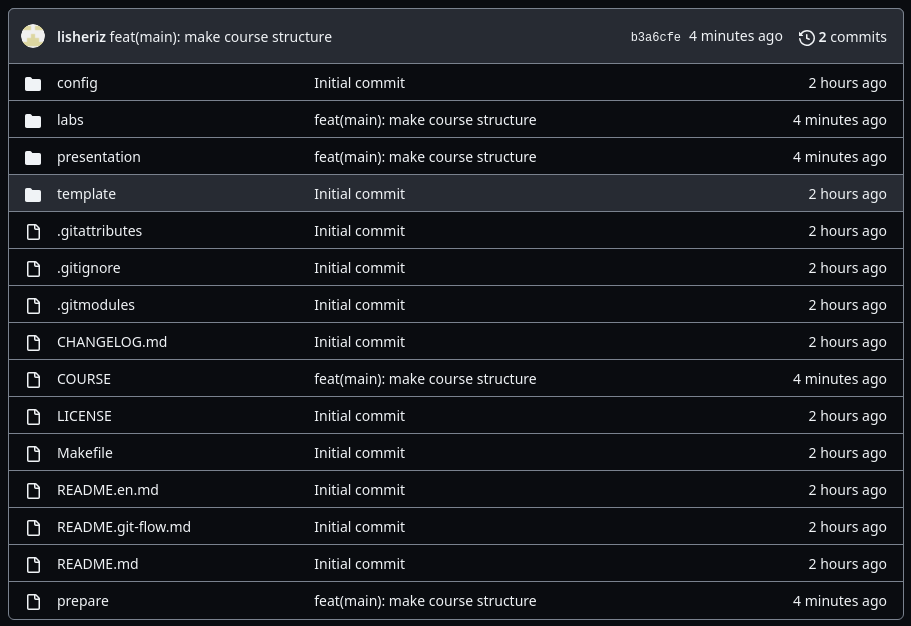


Figure 19: Проверка правильности создания иерархии

(рис. [24](#fig:001)).

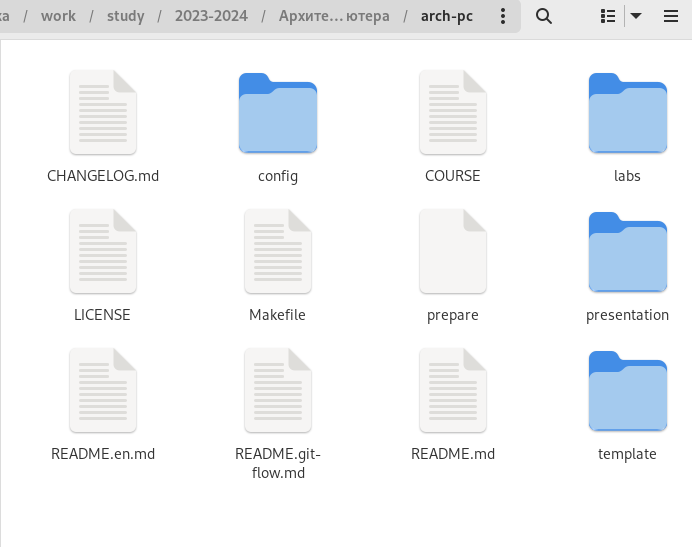


Figure 20: Проверка правильности создания иерархии

## 4.7 **Задания для самостоятельной работы**

1. Создаю отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs/lab02/report). (рис. [24](#fig:001)).

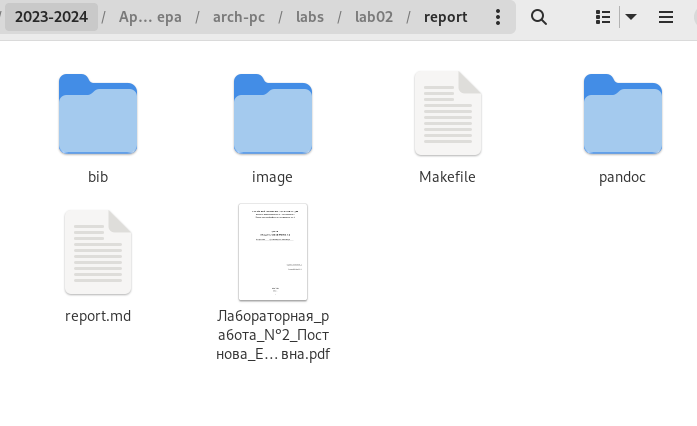


Figure 21: Создание отчета в каталоге

1. Копирую отчет по выполнению предыдущей лабораторной работы в соответствующий каталог созданного рабочего пространства. (рис. [24](#fig:001)).

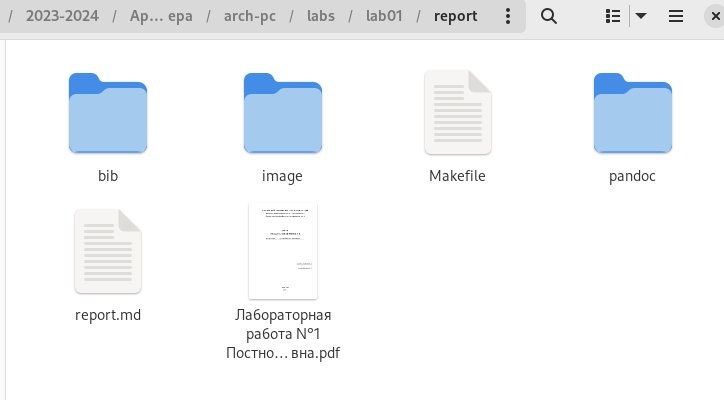


Figure 22: Копирование отчета в каталог

1. Загружаю файлы на github. (рис. [24](#fig:001)).

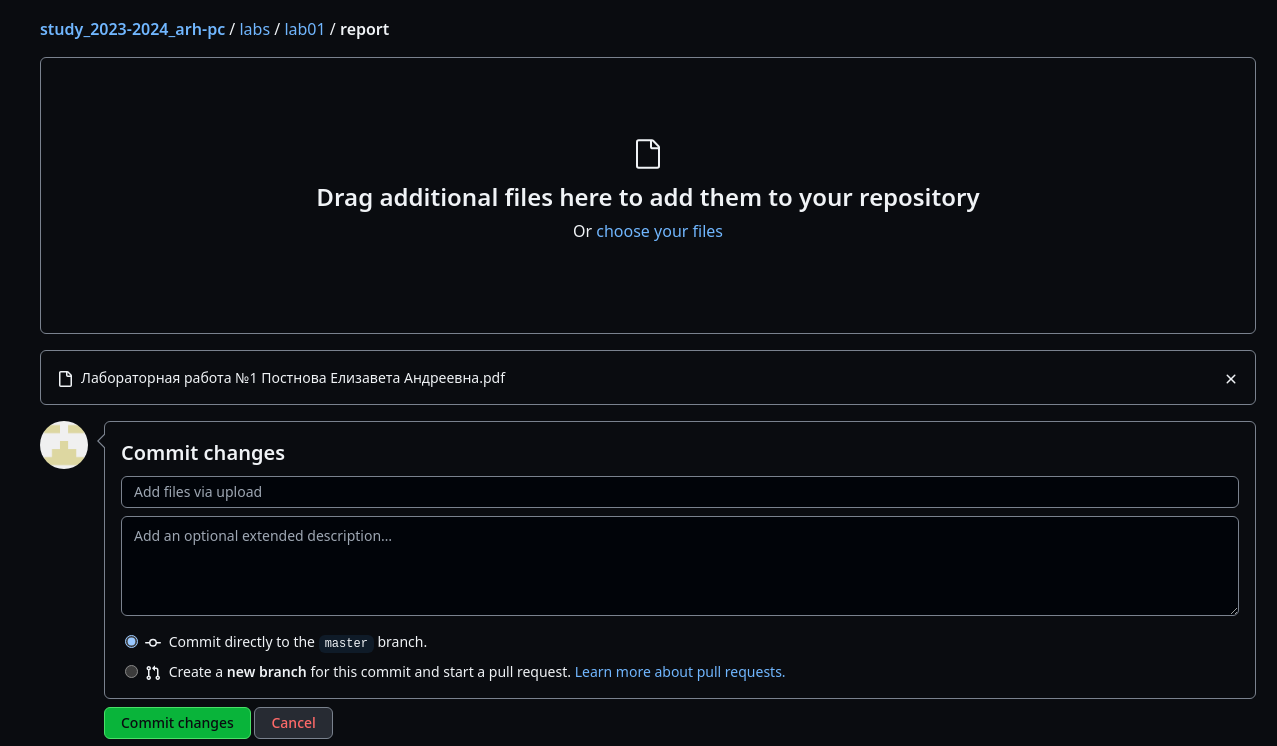


Figure 23: Загрузка файлов лабораторных работ на github

(рис. [24](#fig:001)).

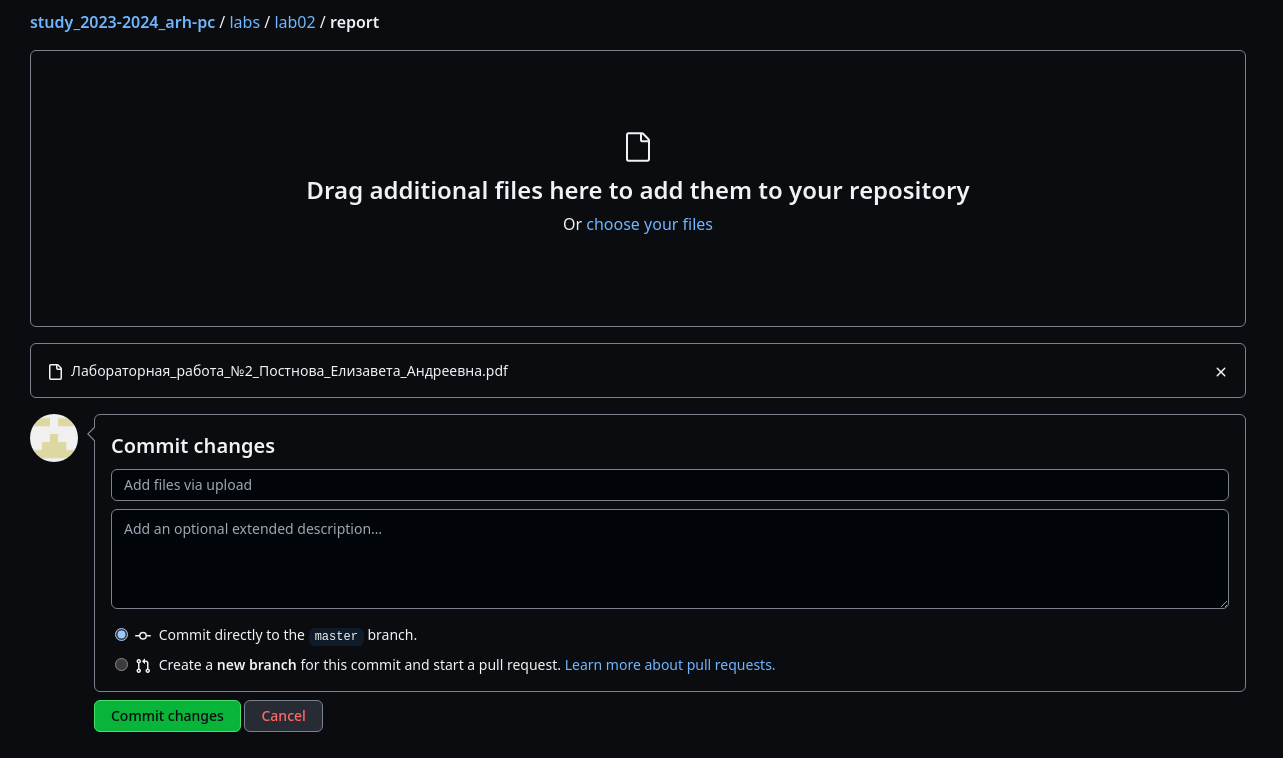


Figure 24: Загрузка файлов лабораторных работ на github

# 5 Выводы

По результатам данной лабораторной работы я приобрела практические навыки по работе с системой git, изучила идеологию и применение средств контроля версий. Эти навыки помогут мне в дальнейшем в работе с данной системой.

# 6 Список литературы

1. GDB: The GNU Project Debugger. — URL: https://www.gnu.org/software/gdb/.
2. GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/.
3. Midnight Commander Development Center. — 2021. — URL: https://midnight-commander. org/.
4. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O’Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658.
6. Robbins A. Bash Pocket Reference. — O’Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: https://www.nasm.us/docs.php.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон- Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: https://www.opennet.ru/docs/RUS/nasm/.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm\_unix.
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).