

Отчёт по лабораторной работе №2

Дисциплина: Архитектура компьютеров

Карпова Анастасия Александровна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	Создание учётной записи на GitHub.	9
4.2	Выполнение предварительной конфигурации git	9
4.3	Настройка кодировки	9
4.4	Создание имени начальной ветки	10
4.5	Параметры autocrlf и safecrlf	10
4.6	Генерация SSH-ключа	10
4.7	Установка утилиты xclip.	11
4.8	Копирование содержимого файла.	11
4.9	Выбор SSH and GPG keys.	12
4.10	Создание нового ключа.	12
4.11	Создание SSH-ключа.	13
4.12	Создание рабочего пространства.	13
4.13	Выбор шаблона.	13
4.14	Создание репозитория.	14
4.15	Перемещение в каталог курса.	14
4.16	Копирование ссылки для клонирования.	14
4.17	Клонирование репозитория.. . . .	15
4.18	Удаление лишних файлов.	15
4.19	Создание необходимых каталогов.	15
4.20	Отправка файлов на сервер.	15
4.21	Проверка.	16
4.22	Создание файла.	16
4.23	Перемещение между каталогами.	16
4.24	Проверка.	16
4.25	Копирование файла	17
4.26	Копирование файла.	17
4.27	Добавление файлов на сервер.	17
4.28	Сохранение изменений.	17
4.29	Отправка сохраненных изменений.	17

Список таблиц

1 Цель работы

Целью данной работы является изучение идеологии и применение средств контроля версий. Приобретение практических навыков 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. Задание 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. 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. 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.

4 Выполнение лабораторной работы

Настройка GitHub

Создаю учётную запись на GitHub (рис. 4.1).

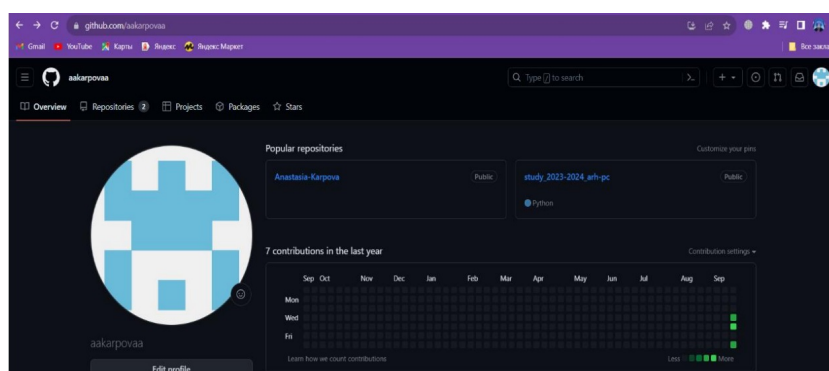


Рис. 4.1: Создание учётной записи на GitHub.

Базовая настройка git Открываю терминал для дальнейшей работы. Выполняю предварительную конфигурацию git: открыв терминал ввожу команды `git config --global user.name ""` и `git config --global user.email "work@mail"`, указывая имя и email владельца репозитория

```
akarpova@Justclown:~$ git config --global user.name "<Anastasia Karpova>"
akarpova@Justclown:~$ git config --global user.email "<1132231934@pfur.ru>"
```

Рис. 4.2: Выполнение предварительной конфигурации git

Настраиваю utf-8 в выводе сообщений git. (рис. 4.3)

```
akarpova@Justclown:~$ git config --global core.quotePath false
```

Рис. 4.3: Настройка кодировки

Задаю имя начальной ветки(будем называть ее master). (рис. 4.4)

```
akarpova@Justclown:~$ git config --global init.defaultBranch master
```

Рис. 4.4: Создание имени начальной ветки

Задаю параметр autocrlf и параметр salecrlf со значениями input и warn соответственно. (рис. ??)

```
akarpova@Justclown:~$ git config --global core.autocrlf input
akarpova@Justclown:~$ git config --global core.safecrlf warn
```

Рис. 4.5: Параметры autocrlf и safecrlf

Создание SSH ключа

Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый). Для этого использую команду `ssh-keygen -C "Имя Фамилия work@mail"`. Ключи сохраняются в каталоге `~/.ssh/`.

```
akarpova@Justclown:~$ ssh-keygen -C "Anastasia Karpova <1132231934@pfur.rudn>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/akarpova/.ssh/id_rsa):
Created directory '/home/akarpova/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/akarpova/.ssh/id_rsa
Your public key has been saved in /home/akarpova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:JqKnay70a0LJ7iJooza+wgjTgjIQzLA0kJCblBedT24 Anastasia Karpova <1132231934@pfur.rudn>
The key's randomart image is:
+---[RSA 3072]-----+
|o..o .
|+o.. o .
|. * . +
|= . E
|+ . o S
|. + . o
|Bo= .
|/X++
|^#B.
+---[SHA256]-----+
```

Рис. 4.6: Генерация SSH-ключа

Далее необходимо загрузить сгенерённый открытый ключ. Для этого зайти на сайт <http://github.org/> под своей учётной записью и перейти в меню Setting. После этого выбрать в боковом меню SSH and GPG keys и нажать кнопку New SSH key. Скопировав из локальной консоли ключ в буфер обмена при помощи

команды `cat ~/.ssh/id_rsa.pub | xclip -sel clip` вставляю ключ в появившееся на сайте поле и указываем для ключа имя. Обратим внимание на команду, где `xclip` – утилита для копирования любого текста через терминал. Но изначально она не установлена в дистрибутиве, поэтому для дальнейшей работы устанавливаем при помощи команды `apt-get install` с ключом `-y`, при этом в начале команды использую `sudo` (для установки от имени админа. (рис. 4.7)

```
akarpova@Justclown:~$ sudo apt install xclip
[sudo] password for akarpova:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
debconf: (Dialog frontend requires a screen at least 13 lines tall and 31 columns wide.)
debconf: falling back to frontend: Readline
Selecting previously unselected package xclip.
(Reading database ... 334031 files and directories currently installed.)
Preparing to unpack .../xclip_0.13-2_amd64.deb ...
Unpacking xclip (0.13-2) ...
Setting up xclip (0.13-2) ...
```

Рис. 4.7: Установка утилиты `xclip`.

После установки утилиты `xclip` мы можем воспользоваться командой `cat ~/.ssh/id_rsa.pub | xclip -sel clip` (рис. 4.8)

```
akarpova@Justclown:~$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
akarpova@Justclown:~$
```

Рис. 4.8: Копирование содержимого файла.

Теперь захожу в свой профиль и выбираю категорию SSH and GPG keys. (рис. 4.9)

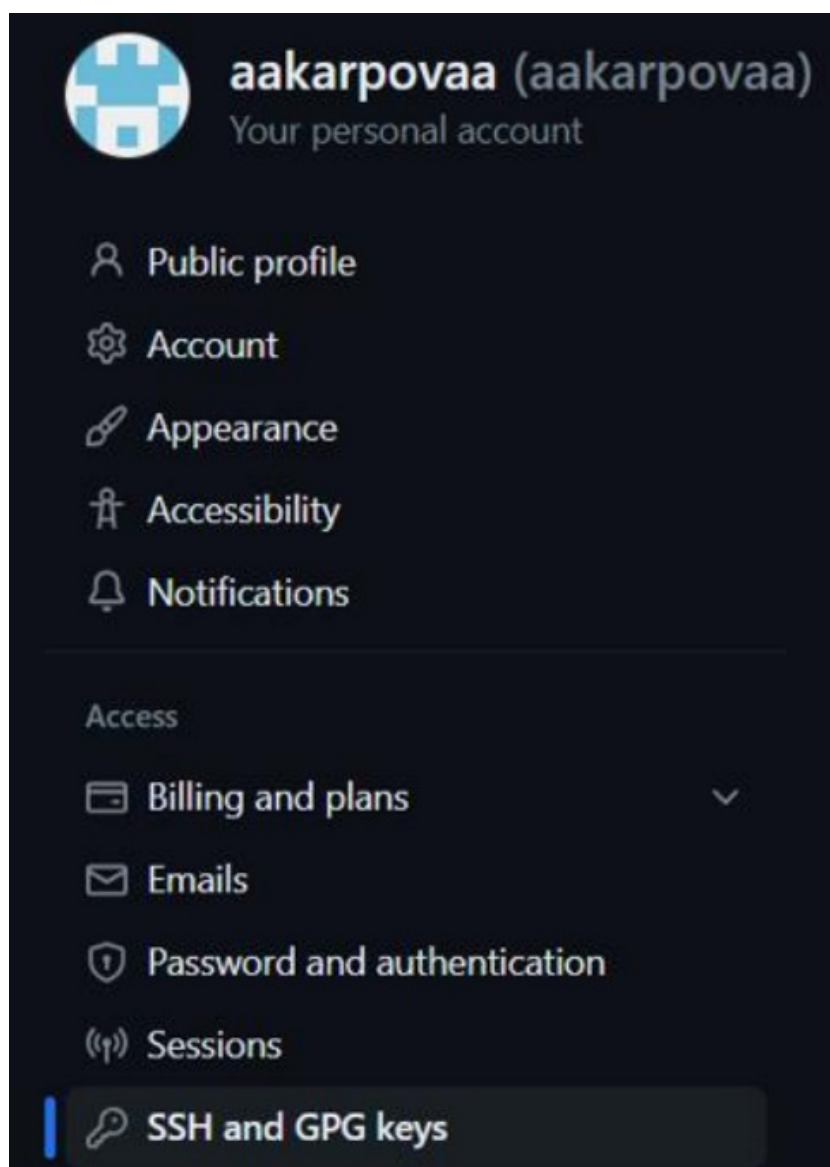


Рис. 4.9: Выбор SSH and GPG keys.

Далее нажимаю на кнопку new SSH key и создаю новый ключ. (рис. 4.10)

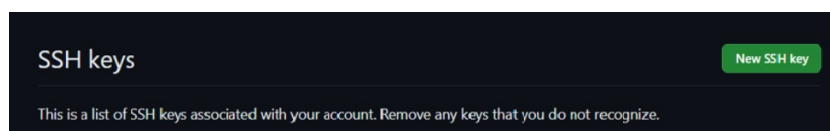


Рис. 4.10: Создание нового ключа.

Вставляю скопированный ключ в поле “key”. В поле Title указываю имя ключа. Нажимаю Add SSH-key, чтобы добавить ключ. (рис. 4.11)

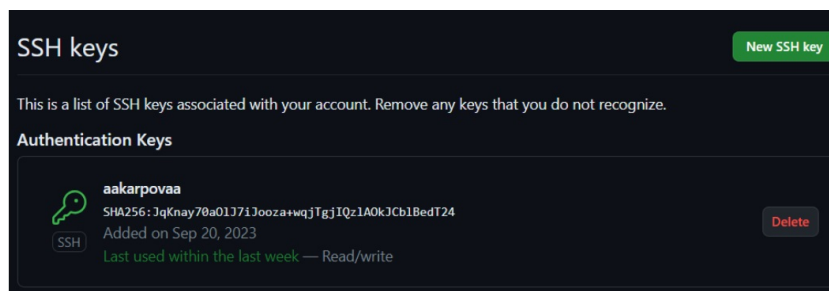


Рис. 4.11: Создание SSH-ключа.

Создание рабочего пространства и репозитория курса на основе шаблона
Создаю директорию и рабочее пространство с помощью утилиты mkdir, благодаря ключу -p создаю все директории после домашней ~/work/study/2023-2024/”Архитектура компьютера” рекурсивно. Для проверки использую ls. (рис. 4.12)

```
aakarpova@Justclown: ~$ mkdir -p work/study/2023-2024/"Архитектура компьютера"
aakarpova@Justclown: ~$ ls
Documents Downloads Images Photos Videos work
```

Рис. 4.12: Создание рабочего пространства.

Создание репозитория курса на основе шаблона
Перехожу на страницу репозитория с шаблоном курса введя адрес <https://github.com/yamadharma/course-directory-student-template> в браузерной строке. Далее выбираю Use this template. (рис. 4.13)

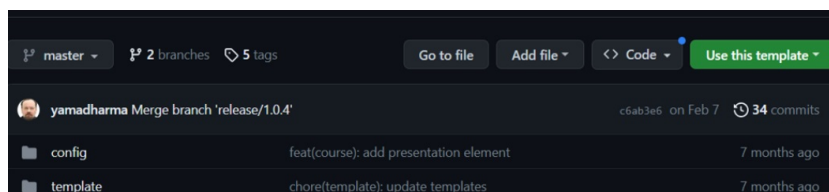


Рис. 4.13: Выбор шаблона.

В открывшемся окне задаю имя репозитория (Repository name) study_2023–2024_arhpc и создайте репозиторий (кнопка Create repository from template). (рис. 4.14)

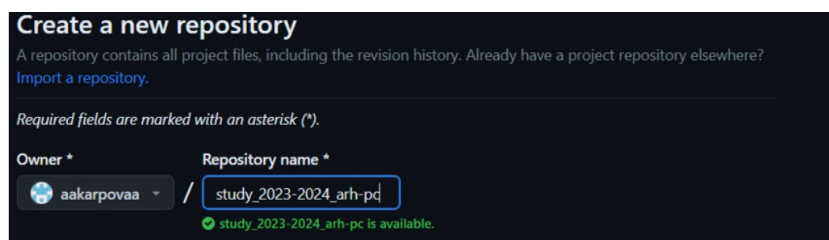


Рис. 4.14: Создание репозитория.

Открываю терминал и перехожу в каталог курса. (рис. ??)

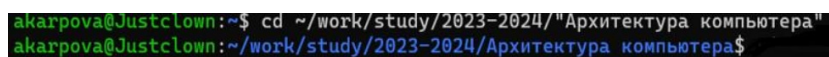


Рис. 4.15: Перемещение в каталог курса.

Клонирую созданный репозиторий (рис. 4.17), предварительно копируя ссылку для клонирования, которую можно найти на странице созданного репозитория. (рис. 4.16)

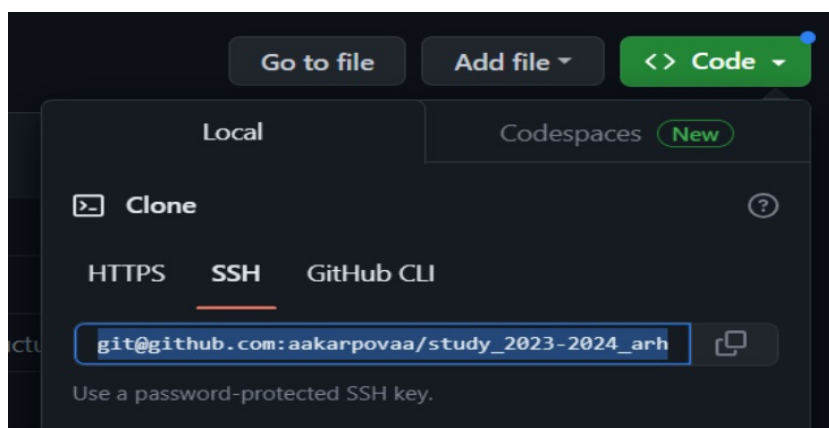


Рис. 4.16: Копирование ссылки для клонирования.

```

akarpova@JustCLown:~/work/study/2023-2024/Архитектура компьютера$ git clone --recursive git@github.com:aakarpovaa/study_
2023-2024_arch-pc.git arch-pc
Cloning into 'arch-pc'...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 27 (delta 1), reused 11 (delta 0), pack-reused 0
Receiving objects: 100% (27/27), 16.93 KiB | 184.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
Submodule 'template/presentation' (https://github.com/yamadharma/academic-presentation-markdown-template.git) registered
for path 'template/presentation'
Submodule 'template/report' (https://github.com/yamadharma/academic-laboratory-report-template.git) registered for path

```

Рис. 4.17: Клонирование репозитория..

Настройка каталога курса

Перехожу в каталог курса при помощи `cd` и удаляю лишние файлы при помощи команды `rm package.json`. (рис. 4.18)

```

akarpova@JustCLown:~/work/study/2023-2024/Архитектура компьютера$ cd ~/work/study/2023-2024/Архитектура компьютера/arch-
pc
akarpova@JustCLown:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ rm package.json

```

Рис. 4.18: Удаление лишних файлов.

Создаю необходимые каталоги(рис. 4.19). И отправляю файлы на сервер(рис. 4.20).

```

akarpova@JustCLown:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ echo arch-pc > COURSE
akarpova@JustCLown:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ make

```

Рис. 4.19: Создание необходимых каталогов.

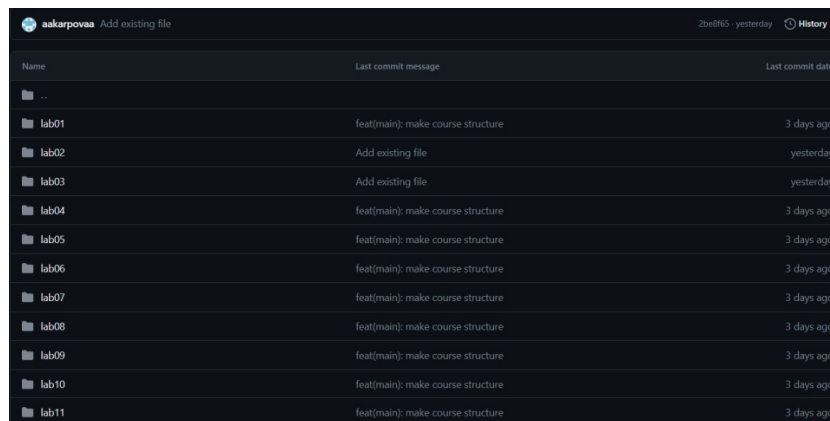
```

akarpova@JustCLown:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git add .
akarpova@JustCLown:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git commit -am 'feat(main): make course struct
ure'
[master 7cfc6c] feat(main): make course structure
199 files changed, 54725 insertions(+), 14 deletions(-)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
akarpova@JustCLown:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git push

```

Рис. 4.20: Отправка файлов на сервер.

Так как я забыла сделать скриншот результата команды `git commit` в терминале, я прикрепляю итог использования данной команды. (рис. 4.21)



Name	Last commit message	Last commit date
..		
lab01	feat(main): make course structure	3 days ago
lab02	Add existing file	yesterday
lab03	Add existing file	yesterday
lab04	feat(main): make course structure	3 days ago
lab05	feat(main): make course structure	3 days ago
lab06	feat(main): make course structure	3 days ago
lab07	feat(main): make course structure	3 days ago
lab08	feat(main): make course structure	3 days ago
lab09	feat(main): make course structure	3 days ago
lab10	feat(main): make course structure	3 days ago
lab11	feat(main): make course structure	3 days ago

Рис. 4.21: Проверка.

Выполнение заданий для самостоятельной работы.

1. Перехожу в директорию labs/lab02/report при помощи cd и создаю в каталоге файл для отчета по второй лабораторной работе. (рис. 4.22)

```
akarpova@Justclown:~$ cd ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report
akarpova@Justclown:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report$ touch Л02_Карпова_отчет
akarpova@Justclown:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report$
```

Рис. 4.22: Создание файла.

2. Перехожу из подкаталога lab02/report в подкаталог lab01/report. (рис. 4.23)

```
akarpova@Justclown:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report$ cd ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab01/report
akarpova@Justclown:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab01/report$
```

Рис. 4.23: Перемещение между каталогами.

Проверяю местонахождение файлов с отчетами по первой и второй лабораторным работам, используя команду ls. (рис. 4.24)

```
akarpova@Justclown:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab01/report$ ls ~/Downloads
LibreOffice_7.6.1_Win_x86-64.msi Л01_Карпова_отчет.pdf Л02_Карпова_отчет.pdf
akarpova@Justclown:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab01/report$ |
```

Рис. 4.24: Проверка.

Копирую первую лабораторную работу при помощи `cp` и проверяю правильность выполнения, используя `ls`. (рис. 4.25)

```
akarpova@Justclown:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab01/report$ cp ~/Downloads/Л01_Карпова_отчет.pdf /home/akarpova/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab01/report
akarpova@Justclown:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab01/report$ ls
Downloads  Makefile  bib       image    pandoc   report.md  Л01_Карпова_отчет.pdf
```

Рис. 4.25: Копирование файла

Перехожу из подкаталог `lab02/report` в подкаталог `lab01/report`. И копирую вторую работу аналогично первой. (рис. 4.26)

```
akarpova@Justclown:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab01/report$ cd ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report
akarpova@Justclown:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report$ cp ~/Downloads/Л02_Карпова_отчет.pdf /home/akarpova/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report
akarpova@Justclown:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report$
```

Рис. 4.26: Копирование файла.

Добавляю файл `Л01_Карпова отчет.pdf` и файл `Л02_Карпова отчет`, используя `git add`. (рис. 4.27) Сохраняю изменения на сервере при помощи команды `git commit -m`. (рис. 4.28)

```
akarpova@Justclown:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab01/report$ git add Л01_Карпова_отчет.pdf
akarpova@Justclown:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab01/report$ cd ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report
akarpova@Justclown:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report$ git add Л02_Карпова_отчет
```

Рис. 4.27: Добавление файлов на сервер.

```
akarpova@Justclown:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report$ git commit -m "Add existing file"
[master 3d58847] Add existing file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 labs/lab02/report/Л02_Карпова_отчет
```

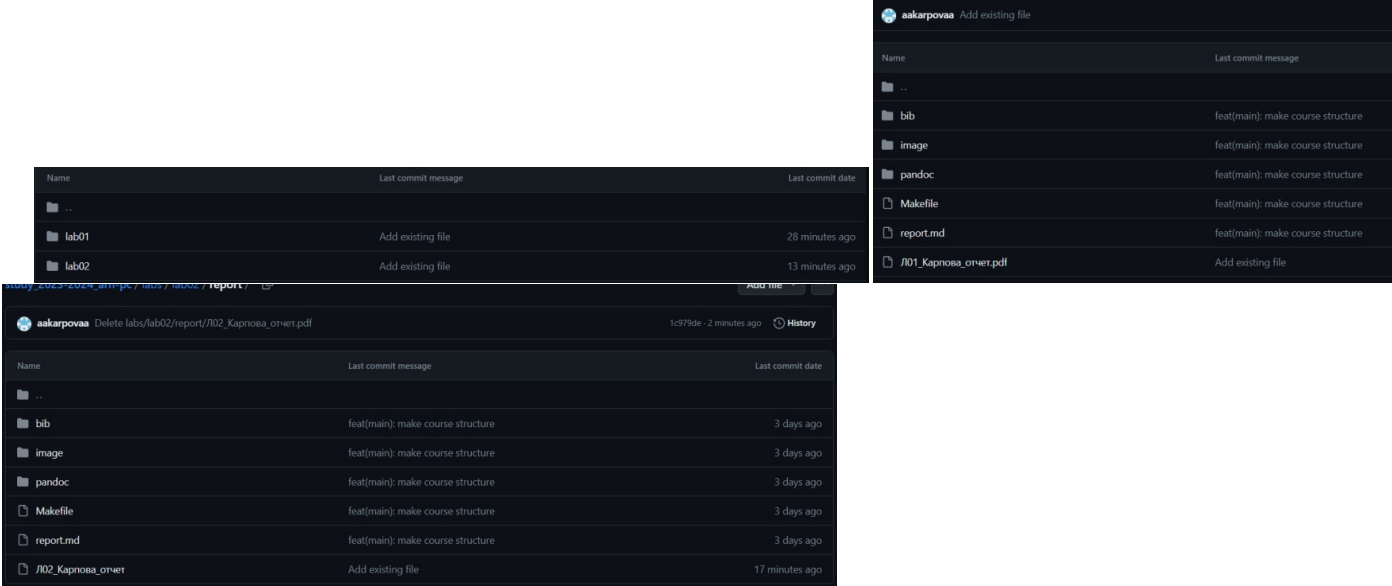
Рис. 4.28: Сохранение изменений.

Отправляю в центральный репозиторий сохраненные изменения. (рис. 4.29)

```
akarpova@Justclown:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab02/report$ git push -f origin master
Enumerating objects: 84, done.
Counting objects: 100% (84/84), done.
Delta compression using up to 4 threads
Compressing objects: 100% (76/76), done.
Writing objects: 100% (84/84), 877.21 KiB | 5.77 MiB/s, done.
Total 84 (delta 17), reused 26 (delta 1), pack-reused 0
remote: Resolving deltas: 100% (17/17), done.
To github.com:aakarpovaa/study_2023-2024_arh-pc.git
+ d92c7ff...3d58847 master -> master (forced update)
```

Рис. 4.29: Отправка сохраненных изменений.

Проверяю выполненные команды. Для этого захожу на github. (рис. ??) (рис. ??) (рис. ??)



5 Выводы

В ходе лабораторной работы я изучила идеологию и применение средств контроля версий и приобрела практические навыки по работе с системой git.

Список литературы

1. Архитектура ЭВМ
2. Инструкция по использованию git ::: {#refs} :::