

# Отчёт по лабораторной работе №3

Дисциплина: архитектура компьютера

Саммура Халед

Нкабд-05-24

1032239384

# Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Настройка GitHub	9
4.2	Базовая настройка Git	10
4.3	Создание SSH-ключа	11
4.4	Создание рабочего пространства и репозитория курса на основе шаблона	14
4.5	Создание репозитория курса на основе шаблона	15
4.6	Настройка каталога курса	17
4.7	Выполнение заданий для самостоятельной работы	20
5	Выводы	27
6	Список литературы	28

## Список иллюстраций

4.1	Заполнение данных учетной записи GitHub	Error! Bookmark not defined.
4.2	Аккаунт GitHub	8
4.3	Предварительная конфигурация git	8
4.4	Настройка кодировки	9
4.5	Создание имени для начальной ветки	9
4.6	Параметр autocrlf	9
4.7	Параметр safecrlf	9
4.8	Генерация SSH-ключа	10
4.9	Установка утилиты xclip	Error! Bookmark not defined.

4.10 Копирование содержимого файла.....	10
4.11 Окно SSH and GPG keys.....	11
4.12 Добавление ключа.....	11
4.13 Создание рабочего пространства.....	11
4.14 Страница шаблона для репозитория.....	12
4.15 Окно создания репозитория.....	12
4.16 Созданный репозиторий.....	13
4.17 Перемещение между директориями.....	13
4.18 Клонирование репозитория.....	14
4.19 Окно с ссылкой для копирования репозитория.....	14
4.20 Перемещение между директориями.....	14
4.21 Удаление файлов.....	15
4.22 Создание каталогов.....	15
4.23 Добавление и сохранение изменений на сервере.....	15
4.24 Выгрузка изменений на сервер.....	16
4.25 Страница репозитория.....	16
4.26 Создание файла.....	<b>Error! Bookmark not defined.</b>
4.27 Меню приложений.....	<b>Error! Bookmark not defined.</b>
4.28 Работа с отчетом в текстовом процессоре.....	<b>Error! Bookmark not defined.</b>
4.29 Перемещение между директориями.....	<b>Error! Bookmark not defined.</b>
4.30 Проверка местонахождения файлов.....	<b>Error! Bookmark not defined.</b>
4.31 Копирование файла.....	<b>Error! Bookmark not defined.</b>
4.32 Перемещение между директориями.....	<b>Error! Bookmark not defined.</b>
4.33 Копирование файла.....	<b>Error! Bookmark not defined.</b>
4.34 Добавление файла на сервер.....	<b>Error! Bookmark not defined.</b>
4.35 Перемещение между директориями.....	<b>Error! Bookmark not defined.</b>
4.36 Добавление файла на сервер.....	<b>Error! Bookmark not defined.</b>

4.37 Подкаталоги и файлы в репозитории .....	Error! Bookmark not defined.
4.38 Отправка в центральный репозиторий сохраненных изменений .....	Error! Bookmark not defined.
4.39 Страница каталога в репозитории.....	Error! Bookmark not defined.
4.40 Страница последних изменений в репозитории .....	Error! Bookmark not defined.
4.41 Каталог lab01/report .....	Error! Bookmark not defined.
4.42 Каталог lab02/report .....	Error! Bookmark not defined.
4.43 Каталог lab03/report .....	Error! Bookmark not defined.

# 1 Цель работы

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

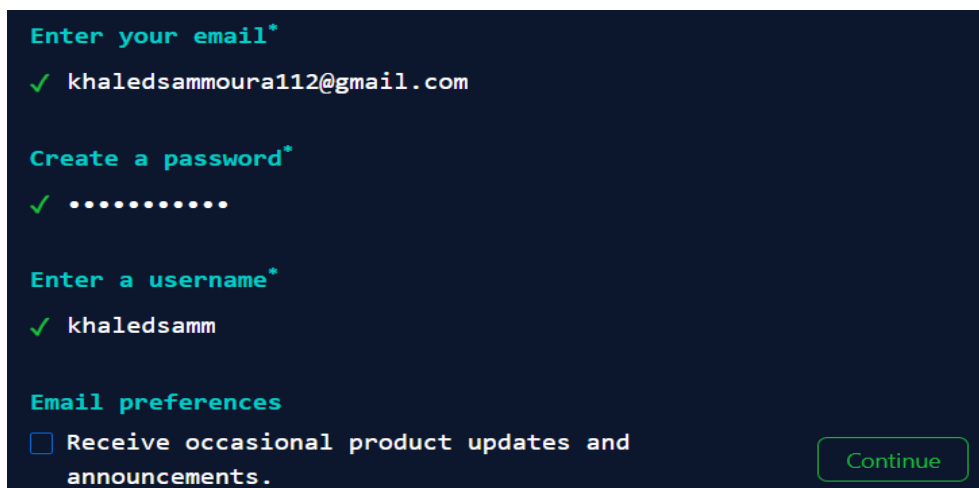
Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля 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 изменения, сделанные разными участниками, 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. 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

Создаю учетную запись на сайте GitHub (рис. 4.1). Далее я заполнила основные данные учетной записи.



The image shows a dark-themed GitHub account creation form. It contains the following fields and elements:

- Enter your email\***: A green checkmark is next to the email address `khaledsammoura112@gmail.com`.
- Create a password\***: A green checkmark is next to a series of dots representing a password.
- Enter a username\***: A green checkmark is next to the username `khaledsamm`.
- Email preferences**: A section with a checkbox labeled `Receive occasional product updates and announcements.` which is currently unchecked.
- Continue**: A green button with white text located at the bottom right of the form.

Рис. 4.1: Заполнение данных учетной записи GitHub

Аккаунт создан (рис. 4.2).

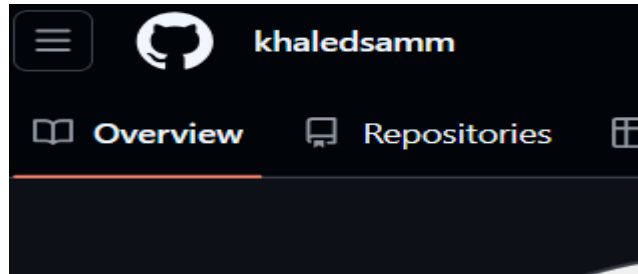


Рис. 4.2: Аккаунт GitHub

## 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"`, указывая в ней электронную почту владельца, то есть мою (рис. 4.3).

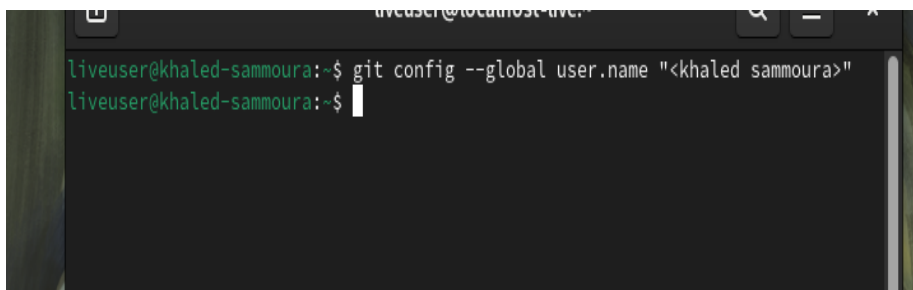


Рис. 4.3: Предварительная конфигурация git

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

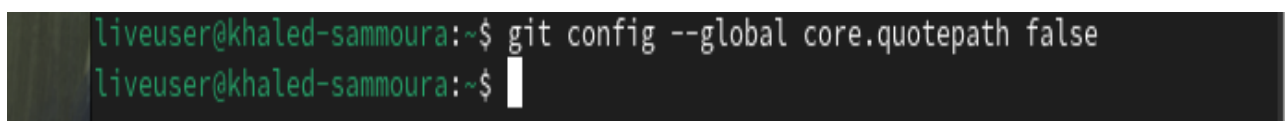




Рис. 4.4: Настройка кодировки

Задаю имя «master» для начальной ветки (рис. 4.5).

```
liveuser@khaled-sammoura:~$ git config --global init.defaultBranch master
liveuser@khaled-sammoura:~$
```

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). CRLF — это символы, которые можно использовать для обозначения разрыва строки в текстовых файлах.

```
liveuser@khaled-sammoura:~$ git config --global core.autocrlf input
liveuser@khaled-sammoura:~$
```

Рис. 4.6: Параметр `autocrlf`

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

```
liveuser@khaled-sammoura:~$ git config --global core.safecrlf warn
liveuser@khaled-sammoura:~$
```

Рис. 4.7: Параметр `safecrlf`

## 4.3 Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозитория 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). Ключ автоматически сохранится в каталоге ~/.ssh/.

```
liveuser@khaledsamoura:~$ ssh-keygen -C "khaled sammoura <khaledsamoura112@gmail.com>"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/liveuser/.ssh/id_ed25519):
Created directory '/home/liveuser/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/liveuser/.ssh/id_ed25519
Your public key has been saved in /home/liveuser/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:aXaTNQyE9b34gP+DATzkNitm+ZcIs9B/1+kYFSuP0vM khaled sammoura <khaledsamoura112@gmail.com>
The key's randomart image is:
+--[ED25519 256]--+
|      ++          |
|     .+.          |
|    + = ..        |
|   .B+ o .o       |
|  .So==o..o       |
| .00..000=        |
| + B o.O.o.       |
|  . + *.B..       |
+-----+-----+
```

Рис. 4.8: Генерация SSH-ключа

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

```
liveuser@khaledsamoura:~$ cat ~/.ssh/id_ed25519.pub
```

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).

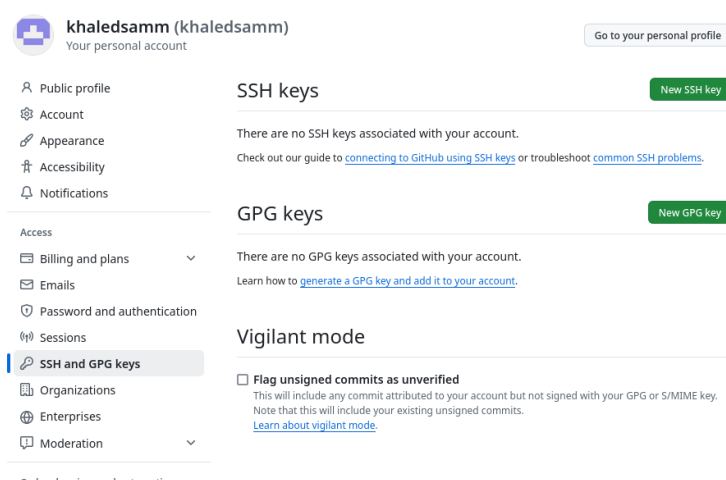


Рис. 4.11: Окно SSH and GPG keys

Вставляю скопированный ключ в поле «Key». В поле Title указываю имя для ключа. Нажимаю «Add SSH-key», чтобы завершить добавление ключа (рис. 4.12).

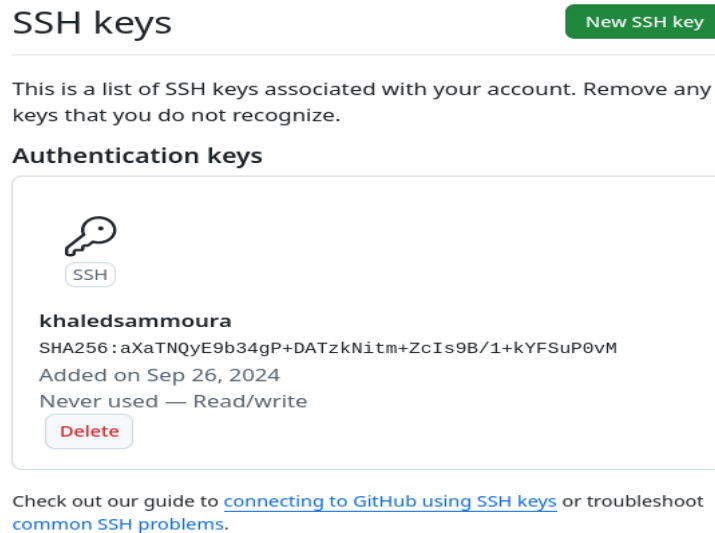


Рис. 4.12: Добавление ключа

## 4.4 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/2022-2023/“Архитектура компьютера” рекурсивно. Далее проверяю с помощью ls, действительно ли были созданы необходимые мне каталоги (рис. 4.13).

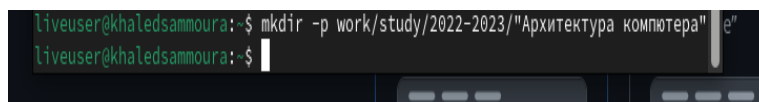


Рис. 4.13: Создание рабочего пространства

## 4.5 Создание репозитория курса на основе шаблона

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

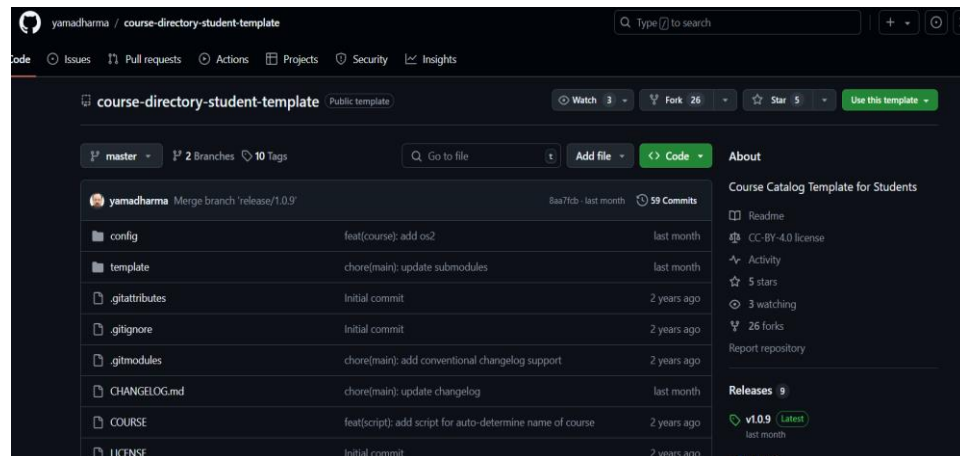


Рис. 4.14: Страница шаблона для репозитория

В открывшемся окне задаю имя репозитория (Repository name): `study_2022–2023_archpc` и создаю репозиторий, нажимая на кнопку «Create repository from template» (рис. 4.15).

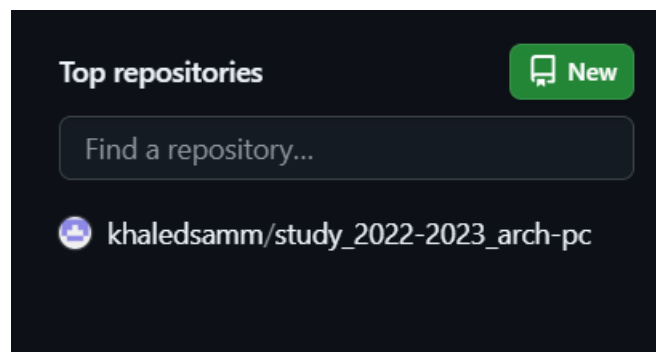


Рис. 4.15: Окно создания репозитория

Репозиторий создан (рис. 4.16).

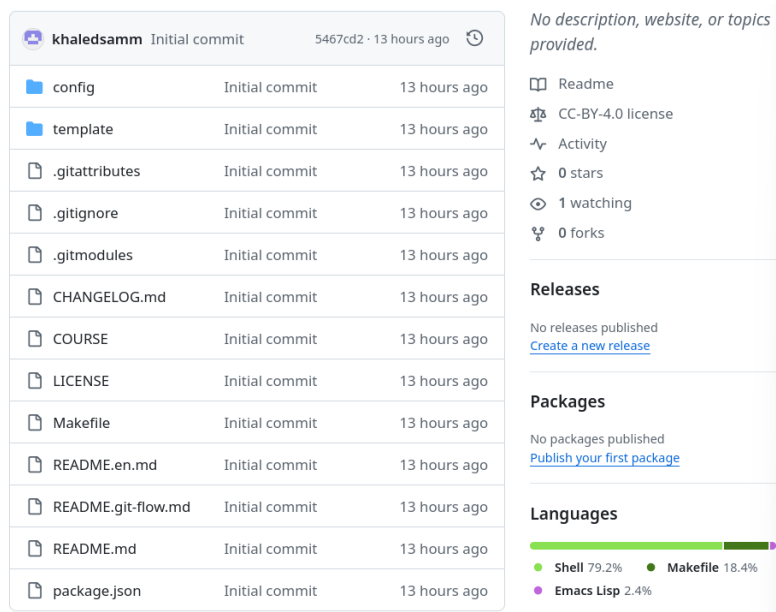


Рис. 4.16: Созданный репозиторий

Через терминал перехожу в созданный каталог курса с помощью утилиты `cd` (рис. 4.17).

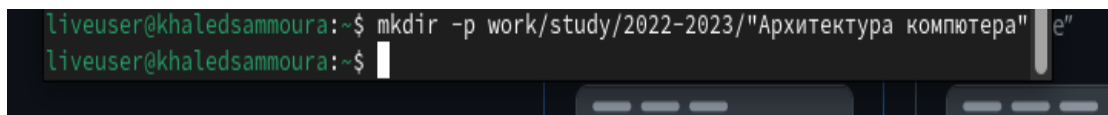


Рис. 4.17: Перемещение между директориями

Клонирую созданный репозиторий с помощью команды `git clone --recursive git@github.com:/study_2022-2023_arh-pc.git arch-pc` (рис. 4.18).

:

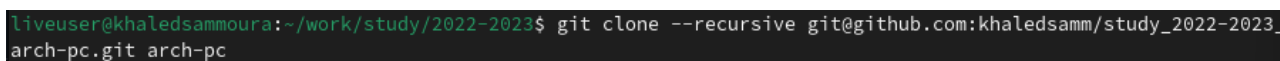
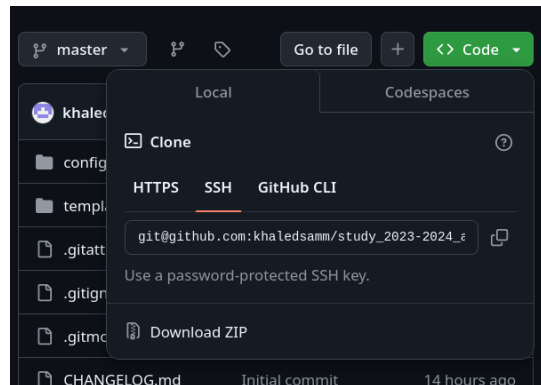


Рис. 4.18: Клонирование репозитория

Копирую ссылку для клонирования на странице созданного репозитория, сначала перейдя в окно «с

ode», далее выбрав в окне вкладку «SSH» (рис. 4.19).



Git-am

Рис. 4.19: Окно с ссылкой для копирования репозитория

## 4.6 Настройка каталога курса

Перехожу в каталог arch-pc с помощью утилиты cd (рис. 4.20).

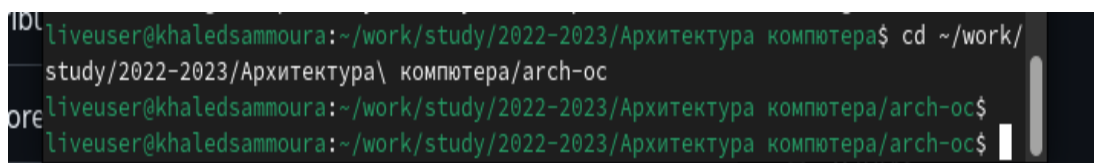


Рис. 4.20: Перемещение между директориями

Удаляю лишние файлы с помощью утилиты rm (рис. 4.21).

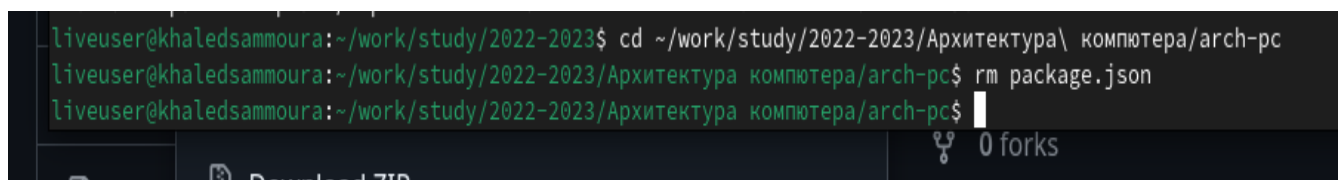


Рис. 4.21: Удаление файлов

Создаю необходимые  
lol каталоги (рис. 4.22).

```
liveuser@khaledsamoura:~/work/study/2022-2023/Архитектура компьютера/arch-oc$ echo arch
-oc > COURSE
liveuser@khaledsamoura:~/work/study/2022-2023/Архитектура компьютера/arch-oc$ make
```

Рис. 4.22: Создание каталогов

Отправляю созданные каталоги 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).

```
liveuser@khaledsamoura:~/work/study/2022-2023/Архитектура компьютера/arch-pc$ git add .
liveuser@khaledsamoura:~/work/study/2022-2023/Архитектура компьютера/arch-pc$ git commit -am 'feat(main): make c
ourse structure'
[master ec22a8e] feat(main): make course structure
223 files changed, 53681 insertions(+), 14 deletions(-)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
create mode 100644 labs/lab01/presentation/.projectile
create mode 100644 labs/lab01/presentation/.texlabroot
create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/image/kulyabov.jpg
create mode 100644 labs/lab01/presentation/presentation.md
create mode 100644 labs/lab01/report/Makefile
create mode 100644 labs/lab01/report/bib/cite.bib
create mode 100644 labs/lab01/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_fignos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_secnos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/core.py
pandoc
```

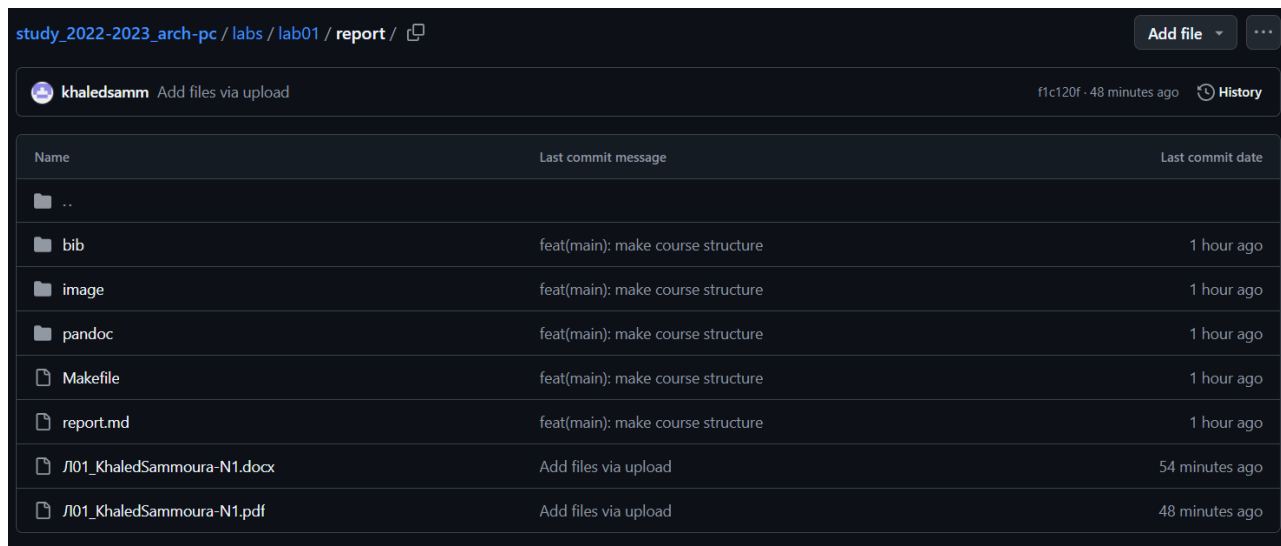
Рис. 4.23: Добавление и сохранение изменений на сервере

Отправляю все на сервер с помощью push (рис. 4.24).

```
liveuser@khaledsamoura:~/work/study/2022-2023/Архитектура компьютера/arch-pc$ git push
Enumerating objects: 37, done.
Counting objects: 100% (37/37), done.
Compressing objects: 100% (29/29), done.
Writing objects: 100% (35/35), 341.40 KiB | 527.00 KiB/s, done.
Total 35 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:khaledsamoura/study_2022-2023_arch-pc.git
5467cd2..ec22a8e master -> master
liveuser@khaledsamoura:~/work/study/2022-2023/Архитектура компьютера/arch-pc$
```

Рис. 4.24: Выгрузка изменений на сервер

Проверяю правильность выполнения работы сначала на самом сайте GitHub  
(рис. 4.25).



study_2022-2023_arch-pc / labs / lab01 / report /			Add file	...
khaledsam Add files via upload			f1c120f · 48 minutes ago	History
Name	Last commit message	Last commit date		
..				
bib	feat(main): make course structure	1 hour ago		
image	feat(main): make course structure	1 hour ago		
pandoc	feat(main): make course structure	1 hour ago		
Makefile	feat(main): make course structure	1 hour ago		
report.md	feat(main): make course structure	1 hour ago		
/l01_KhaledSammoura-N1.docx	Add files via upload	54 minutes ago		
/l01_KhaledSammoura-N1.pdf	Add files via upload	48 minutes ago		

Рис. 4.25: Страница репозитория