

# **Лабораторная работа №2**

**Дисциплина: Архитектура компьютера**

Савостин Олег

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
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 . . . . .	18
4.7	Задания для самостоятельной работы . . . . .	20
<b>5</b>	<b>Выводы</b>	<b>22</b>
	<b>Список литературы</b>	<b>23</b>

# Список иллюстраций

4.1	Сайт github . . . . .	9
4.2	Процесс регистрации на платформе. . . . .	10
4.3	Аккаунт osavostin . . . . .	10
4.4	Команды git . . . . .	11
4.5	Создание ключа. . . . .	11
4.6	Настройки аккаунта osavostin . . . . .	12
4.7	New SSH key . . . . .	12
4.8	Установка пакета файлов xclip и копирка данных ключа . . . . .	13
4.9	Создание нового SSH ключа . . . . .	14
4.10	Создание рабочего пространства . . . . .	15
4.11	Результат перехода по ссылке. . . . .	15
4.12	Create a new repository . . . . .	16
4.13	Копирование репозитории в подкаталог Архитектура компьютера . . . . .	17
4.14	Копирование ссылки. . . . .	18
4.15	rm и echo . . . . .	19
4.16	Отправка файлов на github. . . . .	19
4.17	COURSE . . . . .	20
4.18	Создание отчета и проверка на наличие . . . . .	20
4.19	Копирка файла в lab01/report . . . . .	20
4.20	Загрузка на github . . . . .	21
4.21	labs в github . . . . .	21

## **Список таблиц**

# 1 Цель работы

Целью данной работы является изучить идеологию и применение средств контроля версий и 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. Задания для самостоятельной работы
8. Контрольные вопросы для самопроверки

### 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.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности

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

### 4.1 Настройка github

Для начала пользования сервисом github, я захожу в браузер Firefox, который является основным для Linux и перехожу на сайт <https://www.github.com/> (Рис. 4.1) и заполняю основные данные для регистрации. (Рис.4.2) Аккаунт готов (Рис.4.3).

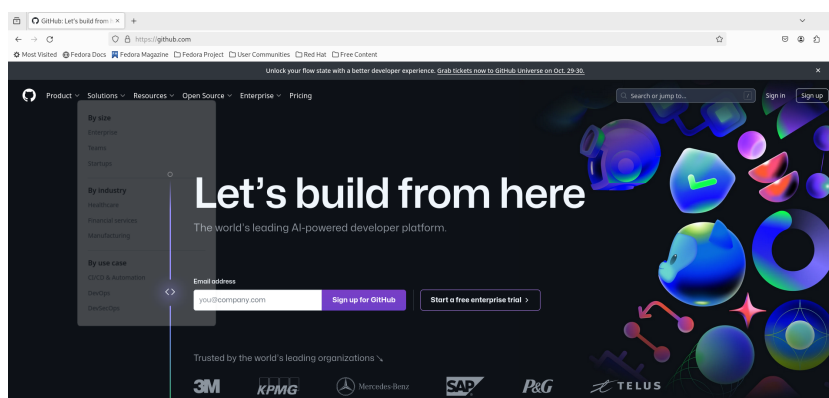


Рис. 4.1: Сайт github

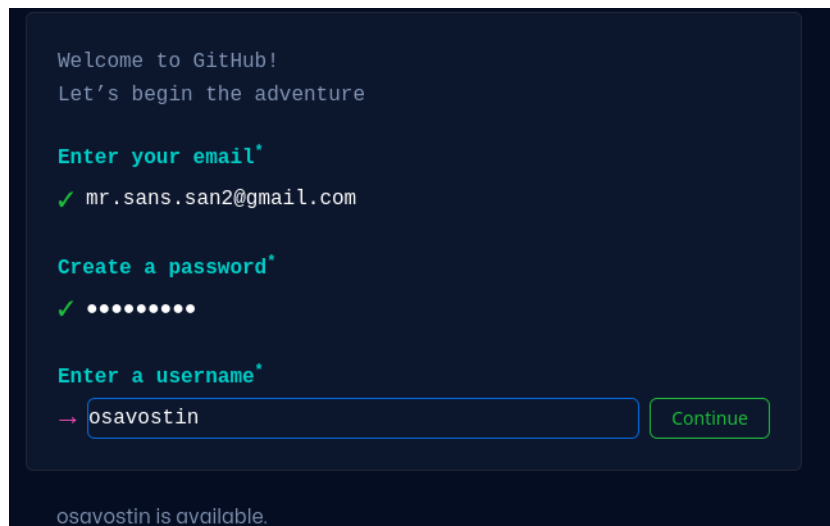


Рис. 4.2: Процесс регистрации на платформе.

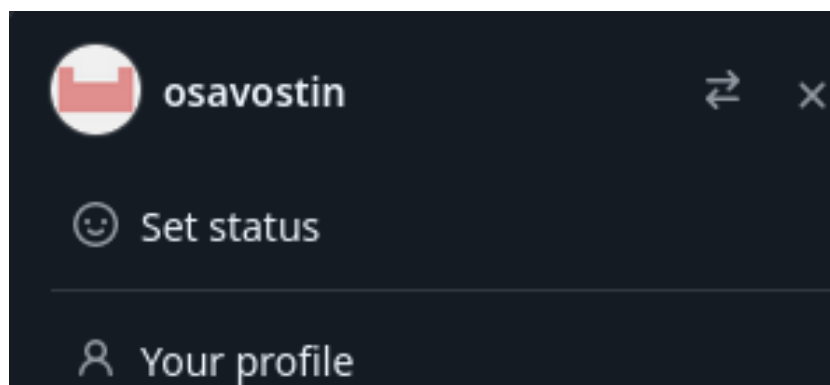


Рис. 4.3: Аккаунт osavostin

## 4.2 Базовая настройка git

Открываю терминал в Linux и ввожу команды `git config --global user.name` и `git config --global user.email`, также указываю свои логин и почту. Затем я настраиваю utf-8 в выводе сообщений git. Задаю имя начальной ветки и будем называть её master. Использую параметры `autocrlf` и `safecrlf` (Рис. 4.4).

```

savostinoleg@vbox:~$ git config --global user.name "osavostin" 1
savostinoleg@vbox:~$ git config --global user.email "mr.sans.san2@gmail.com"
savostinoleg@vbox:~$ git config --global core.quotepath false 2
savostinoleg@vbox:~$ git config --global init.defaultBranch master 3
savostinoleg@vbox:~$ git config --global core.autocrlf input 4
savostinoleg@vbox:~$ git config --global core.safecrlf warn 5
savostinoleg@vbox:~$

```

Рис. 4.4: Команды git

## 4.3 Создание SSH ключа

Для последующей идентификации пользователя (меня) на сервере репозитория я сгенерирую пару ключей (приватный и открытый). Для этого открываю терминал и ввожу команду `ssh-keygen -C "Олег Савостин <mr.sans.san2@gmail.com>"`. Ключ сохраняется в скрытой директории `~/.ssh/` (Рис. 4.5)

```

savostinoleg@vbox:~$ ssh-keygen -C "Олег Савостин <mr.sans.san2@gmail.com>"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/savostinoleg/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/savostinoleg/.ssh/id_ed25519
Your public key has been saved in /home/savostinoleg/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:4QKNFM02SAVbLL0ERft03ft6tX5t/2NDTh+mKUJFvjM Олег Савостин <mr.sans.san2@gmail.com>
The key's randomart image is:
+--[ED25519 256]--+
|  .=%* .          |
|  .o=0.   o  .    |
|  +00+0 +  . .    |
|  ..+ o o   .     |
|  . S . . .      |
|  . . E  * .      |
|  .  o B B       |
|  . . o B*       |
|  . . ++0        |
+----[SHA256]-----+
savostinoleg@vbox:~$

```

Рис. 4.5: Создание ключа.

Чтобы загрузить данный ключ, я захожу на [github.org](https://github.com), захожу в свой аккаунт и захожу в настройки и выбираю опцию SSH and GPG keys (Рис. 4.6)

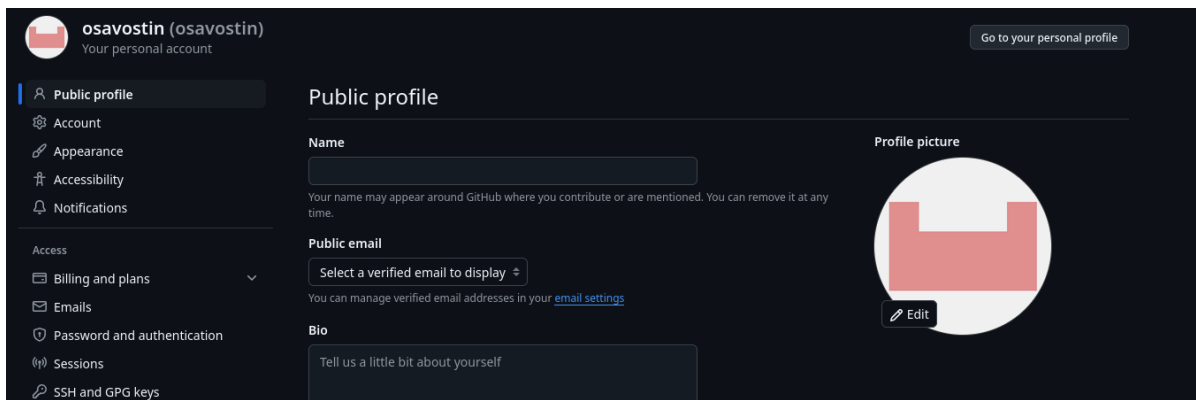


Рис. 4.6: Настройки аккаунта osavostin

Выбираю New SSH key. (Рис. 4.7)

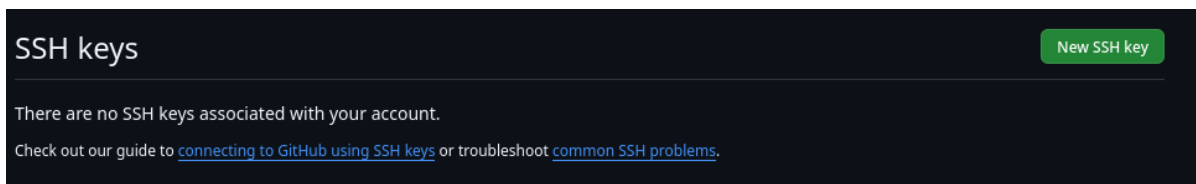


Рис. 4.7: New SSH key

Чтобы скопировать ключ SSH в свой буфер обмена, мне следует установить пакет файлов xclip в терминале. После этого, я ввожу команду `cat ~/.ssh/id_ed25519.pub | xclip -sel clip`, где файл .pub является моим ключом. Я его и копирую. (Рис. 4.8)

```
bash: xclip: команда не найдена...
Установить пакет «xclip», предоставляющий команду «xclip»? [N/y] y

* Ожидание в очереди...
* Загрузка списка пакетов....
Следующие пакеты должны быть установлены:
xclip-0.13-21.git11cba61.fc40.x86_64  Command line clipboard grabber
Продолжить с этими изменениями? [N/y] y

* Ожидание в очереди...
* Ожидание аутентификации...
* Ожидание в очереди...
* Загрузка пакетов...
* Запрос данных...
* Проверка изменений...
* Установка пакетов...

savostinoleg@vbox:~$ cat ~/.ssh/id_ed25519.pub | xclip -sel clip
```

Рис. 4.8: Установка пакета файлов xclip и копирка данных ключа

Теперь, я вставляю скопированные данные в github (Рис. 4.9)

**Add new SSH Key**

**Title**

Новый ключ

**Key type**

Authentication Key

**Key**

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIB0jCExsZsiFRMdEEed8XXQc7/  
udQa2gNXUG72LneDg5i Олег Савостин <mr.sans.san2@gmail.com>
```

Add SSH key

Рис. 4.9: Создание нового SSH ключа

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

Так как при выполнении лабораторных работ следует придерживаться структуры рабочего пространства, я создам каталог для предмета “Архитектура компьютера”(Рис. 4.10)

```
savostinoleg@vbox:~$ mkdir -p ~/work/study/2024-2025/"Архитектура компьютера"
savostinoleg@vbox:~$ ls
work    Документы  Изображения  Общедоступные  Шаблоны
Видео   Загрузки   Музыка        'Рабочий стол'
savostinoleg@vbox:~$
```

Рис. 4.10: Создание рабочего пространства

## 4.5 Создание репозитория курса на основе шаблона

Захожу в web-интерфейс github и перехожу на страницу репозитория с шаблонов <https://github.com/yamadharma/course-directory-student-template> (Рис. 4.11)

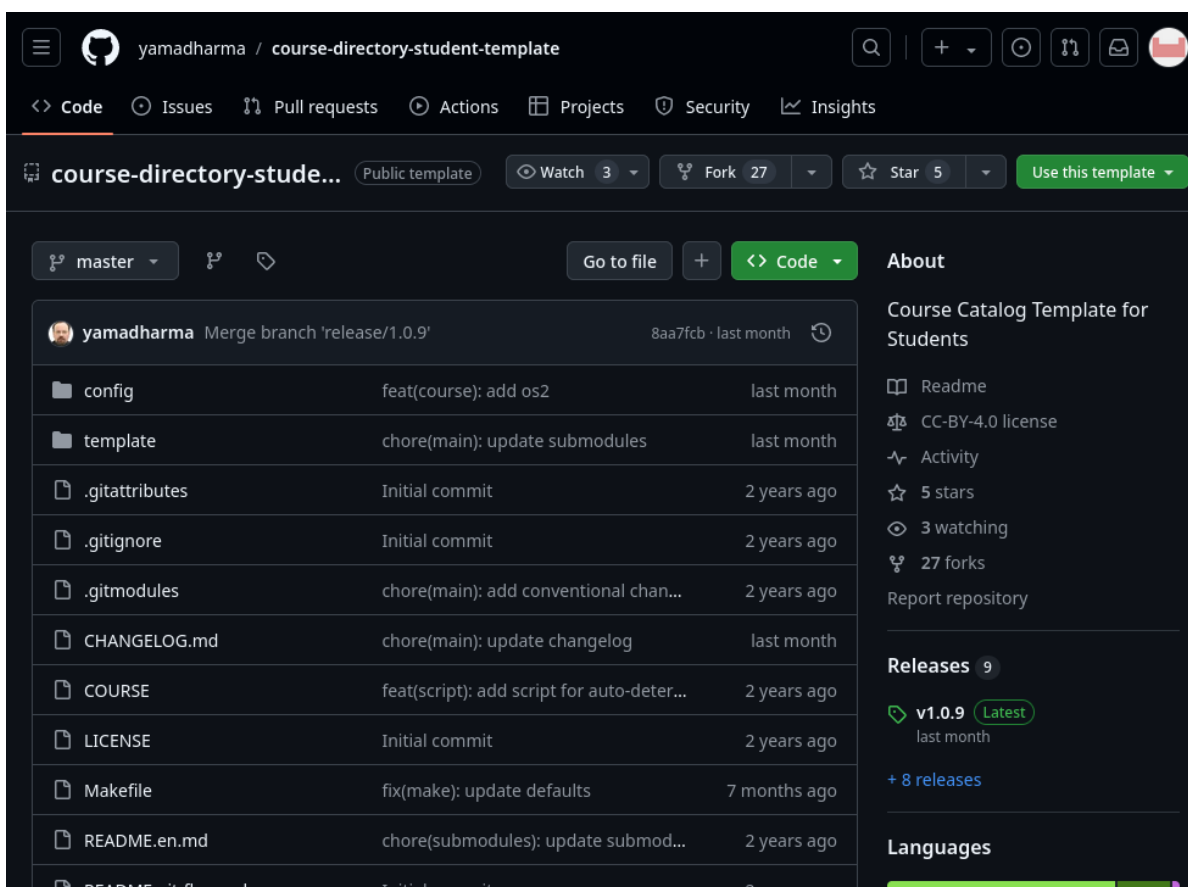


Рис. 4.11: Результат перехода по ссылке.

Теперь использую этот шаблон, нажимая на Use this template и Create new


repository. Затем, задаю ему название study\_2023-2023\_arch-pc и нажимаю Create repository (Рис. 4.12)

**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

*Required fields are marked with an asterisk (\*).*


**Repository template**

 yamadharm/course-directory-student-template ▾

Start your repository with a template repository's contents.

☐ **Include all branches**  
Copy all branches from yamadharm/course-directory-student-template and not just the default branch.


**Owner \*** **Repository name \***


 osavostin ▾ / study\_2023-2024\_arh-pc


✔ study\_2023-2024\_arh-pc is available.

Great repository names are short and memorable. Need inspiration? How about **super-octo-potato** ?

**Description** (optional)

☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

 You are creating a public repository in your personal account.

**Create repository**

Рис. 4.12: Create a new repository

Теперь, я захожу в терминал, и перехожу в каталог “Архитектура компьютера” с помощью команды `~/work/study/2024-2025/”Архитектура компьютера”`. В данный подкаталог я скопирую всё, что было в репозитории с помощью команды `git clone -recursive git@github.com:osavostin/study_2023-2024_arh-pc.git arch-pc` (Рис. 4.13)



```

savostinoleg@vbox: ~/work/study/2024-2025/Архитектура компьютера$ cd ~
savostinoleg@vbox: ~$ cd ~/work/study/2024-2025/"Архитектура компьютера"
savostinoleg@vbox: ~/work/study/2024-2025/Архитектура компьютера$ git clone --recursive git@github.com:osavostin/study_2023-2024_arh-pc.git arch-pc
Клонирование в «study_2023-2024_arh-pc»...
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 33 (delta 1), reused 18 (delta 0), pack-reused 0 (from 0)
Получение объектов: 100% (33/33), 18.81 КиБ | 332.00 КиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharm/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharm/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/savostinoleg/work/study/2024-2025/Архитектура компьютера/study_2023-2024_arh-pc/template/presentation»...
remote: Enumerating objects: 111, done.
remote: Counting objects: 100% (111/111), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 111 (delta 42), reused 100 (delta 31), pack-reused 0 (from 0)
Получение объектов: 100% (111/111), 102.17 КиБ | 500.00 КиБ/с, готово.
Определение изменений: 100% (42/42), готово.
Клонирование в «/home/savostinoleg/work/study/2024-2025/Архитектура компьютера/study_2023-2024_arh-pc/template/report»...
remote: Enumerating objects: 142, done.
remote: Counting objects: 100% (142/142), done.
remote: Compressing objects: 100% (97/97), done.
remote: Total 142 (delta 60), reused 121 (delta 39), pack-reused 0 (from 0)
Получение объектов: 100% (142/142), 341.09 КиБ | 808.00 КиБ/с, готово.
Определение изменений: 100% (60/60), готово.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5086c9c72a02bd2fca1d4a6'
Submodule path 'template/report': checked out 'c26e22effe7b3e0495707d82ef561ab185f5c748'
savostinoleg@vbox: ~/work/study/2024-2025/Архитектура компьютера$

```

Рис. 4.13: Копирование репозитории в подкаталог Архитектура компьютера

Ссылка не была использована в данном случае, но это можно сделать следующим образом (Рис. 4.14)

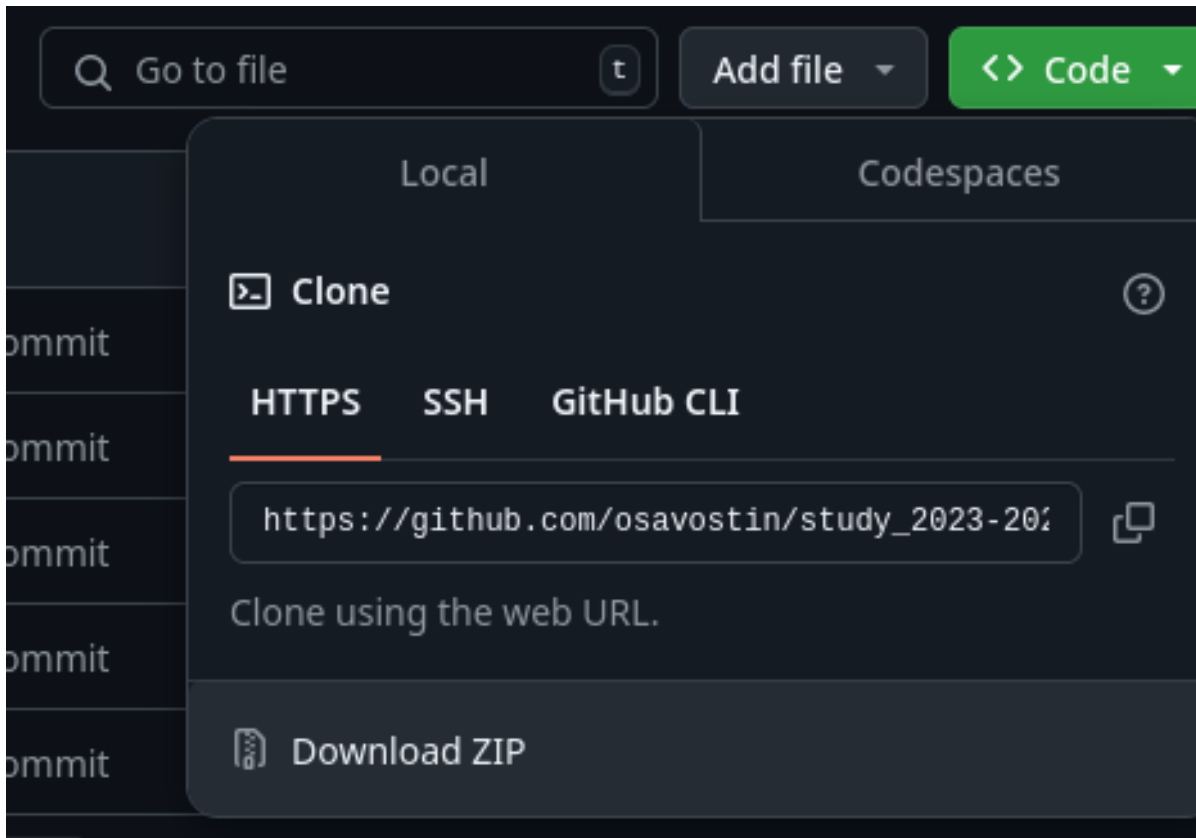


Рис. 4.14: Копирование ссылки.

## 4.6 Настройка каталога курса

Сперва, в терминале я перехожу в каталог курса с помощью утилиты `cd` в каталог `arch-рс` и удаляю лишний файл `package.json` с помощью `rm`, проверяю на его наличие. Затем создаю необходимые каталоги с помощью `echo arch-рс > COURSE`. Устанавливаю пакет `make` и пользуюсь этой командой (Рис. 4.15)

```

savostinoleg@vbox:~$ cd ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ rm package.json
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ ls
CHANGELOG.md  COURSE  Makefile      README.git-flow.md  template
config        LICENSE  README.en.md  README.md
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ echo arch-pc > COURSE
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ make
bash: make: команда не найдена...
Установить пакет «make», предоставляющий команду «make»? [N/y] y

* Ожидание в очереди...
* Загрузка списка пакетов....
Следующие пакеты должны быть установлены:
gc-8.2.2-6.fc40.x86_64 Garbage collector for C and C++
guile30-3.0.7-12.fc40.x86_64 A GNU implementation of Scheme for application extensibility
make-1:4.4.1-6.fc40.x86_64 A GNU tool which simplifies the build process for users
Продолжить с этими изменениями? [N/y] y

* Ожидание в очереди...
* Ожидание аутентификации...
* Ожидание в очереди...
* Загрузка пакетов...
* Запрос данных...
* Проверка изменений...
* Установка пакетов...

```

Рис. 4.15: rm и echo

Теперь отправляю файлы на сервер с помощью git add .; git commit -am; git push. Последняя команда отправляет все файлы на github.(Рис. 4.16)

```

savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git add .
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git commit -am 'feat(main): make course structure'
[master 50fbc38] feat(main): make course structure
 2 files changed, 1 insertion(+), 14 deletions(-)
 delete mode 100644 package.json
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 278 байтов | 55.00 КиБ/с, готово.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:osavostin/study_2023-2024_arh-pc.git
 9afb31b..50fbc38 master -> master
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$

```

Рис. 4.16: Отправка файлов на github.

Проверяю если файл загрузился. (Рис. 4.17)

Рис. 4.17: COURSE

## 4.7 Задания для самостоятельной работы

1. Создаю отчет по выполнению лабораторной работы в labs/lab02/report/ с помощью утилиты touch(Рис. 4.18)

```
savostinoleg@vbox:~$ touch ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab02/report/"Л02_Савостин_отчет.pdf"
savostinoleg@vbox:~$ ls ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab02/report/
Л02_Савостин_отчет.pdf
savostinoleg@vbox:~$
```

Рис. 4.18: Создание отчета и проверка на наличие

2. Копирую предыдущие лабораторные работы в соответствующие им каталоги. Предыдущая работа находилась в каталоге Документы. С помощью cp и ls я копирую данный файл в соответствующий каталог в рабочем пространстве и проверяю на его наличие(Рис. 4.19)

```
savostinoleg@vbox:~$ cp ~/ "Документы"/"Л01_Савостин_отчет.pdf" ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab01/report/
savostinoleg@vbox:~$ ls ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab01/report/
Л01_Савостин_отчет.pdf
savostinoleg@vbox:~$
```

Рис. 4.19: Копирка файла в lab01/report

3. Теперь загружаю файлы на github используя git add .; git commit -m; git push.(Рис. 4.20)

```

savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git add .
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git commit -m "Added lab02 report, its directory and others"
[master dfc3ca8] Added lab02 report, its directory and others
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 labs/lab01/report/Л01_Савостин_отчет.pdf
create mode 100644 labs/lab02/report/Л02_Савостин_отчет.pdf
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (5/5), готово.
Запись объектов: 100% (8/8), 596 байтов | 149.00 КиБ/с, готово.
Total 8 (delta 1), reused 1 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:osavostin/study_2023-2024_arh-pc.git
c8bf354..dfc3ca8 master -> master
savostinoleg@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$

```

Рис. 4.20: Загрузка на github

Как итог, всё прошло успешно. (Рис. 4.21)

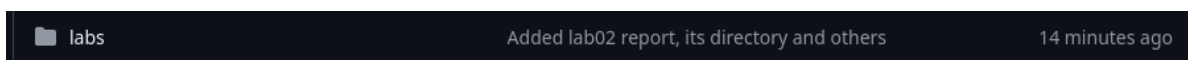


Рис. 4.21: labs в github

## 5 Выводы

При выполнении данной лабораторной работы я изучил применение средств контроля версий, а также приобрел практические навыки по работе с системой git и узнал как пересылать файлы на github.

# Список литературы

Архитектура ЭВМ РУДН