**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

Дисциплина: Архитектура компьютеров

Студент: Поляков

Сергей Александрович

Группа: НБИбд-02-25

**МОСКВА**

2025 г.

**Оглавление**

[1. Цель работы 2](#_Toc1136554349)

[2. Теоретическое введение 2](#_Toc78983391)

[2.1. Системы контроля версий. Общие понятия 2](#_Toc1260990194)

[2.2. Система контроля версий Git 2](#_Toc126869418)

[2.3. Основные команды git 3](#_Toc1223634875)

[2.4. Стандартные процедуры работы при наличии центрального репозитория 3](#_Toc356800104)

[3. Выполнение лабораторной работы 4](#_Toc611917289)

[3.1. Настройка github 5](#_Toc1601066824)

[3.2. Базовая настройка git 5](#_Toc1119653975)

[3.3. Создание SSH-ключа 6](#_Toc418791629)

[3.4. Создание рабочего пространства и репозитория курса на основе шаблонов 9](#_Toc528987861)

[3.5. Создание репозитория курса на основе шаблона 10](#_Toc1676359586)

[3.6. Настройка каталога курса 13](#_Toc1107028722)

[4. Задание для самостоятельной работы 15](#_Toc51251928)

[5. Вывод 20](#_Toc1129272901)

Ссылка на аккаунт GitHub: <https://github.com/zelikovegor03-afk/study_2025-2026_arh-pc.git>

# **1. Цель работы**

Целью работы является изучение идеологии и применения средств контроля версий, приобретение практических навыков по работе с системой контроля версий git.

# **2. Теоретическое введение**

## **2.1. Системы контроля версий. Общие понятия**

Системы контроля версий (VersionControlSystem,VCS) применяются при работе нескольких человек над одни 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.

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.

## **2.2. Система контроля версий Git**

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.

## **2.3. Основные команды git**

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 - добавить конкретные изменённые и/или созданные файлы и/или

каталоги «имена\_файлов».

git rm - удалить файл и/или каталог из индекса репозитория (при этом файл

и/или каталог остаётся в локальной директории) «имена\_файлов».

git commit -am - сохранить все добавленные изменения и все изменённые файлы

«Описание коммита».

git checkout -b - создание новой ветки, базирующейся на текущей «имя\_ветки».

git checkout - переключение на некоторую ветку (при переключении на ветку,

которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)

«имя\_ветки».

git push origin - отправка изменений конкретной ветки в центральный репозиторий

«имя\_ветки».

git merge - слияние ветки с текущим деревом «—no-ff» «имя\_ветки»

git branch -d - удаление локальной уже слитой с основным деревом ветки «имя\_ветки».

git branch -D - принудительное удаление локальной ветки «имя\_ветки».

git push origin - удаление ветки с центрального репозитория «:имя\_ветки».

## **2.4. Стандартные процедуры работы при наличии 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 необ- ходимо разместить их в центральном репозитории. Для этого необходимо проверить, какие файлы изменились к текущему моменту:

git status

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, изме-нения 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

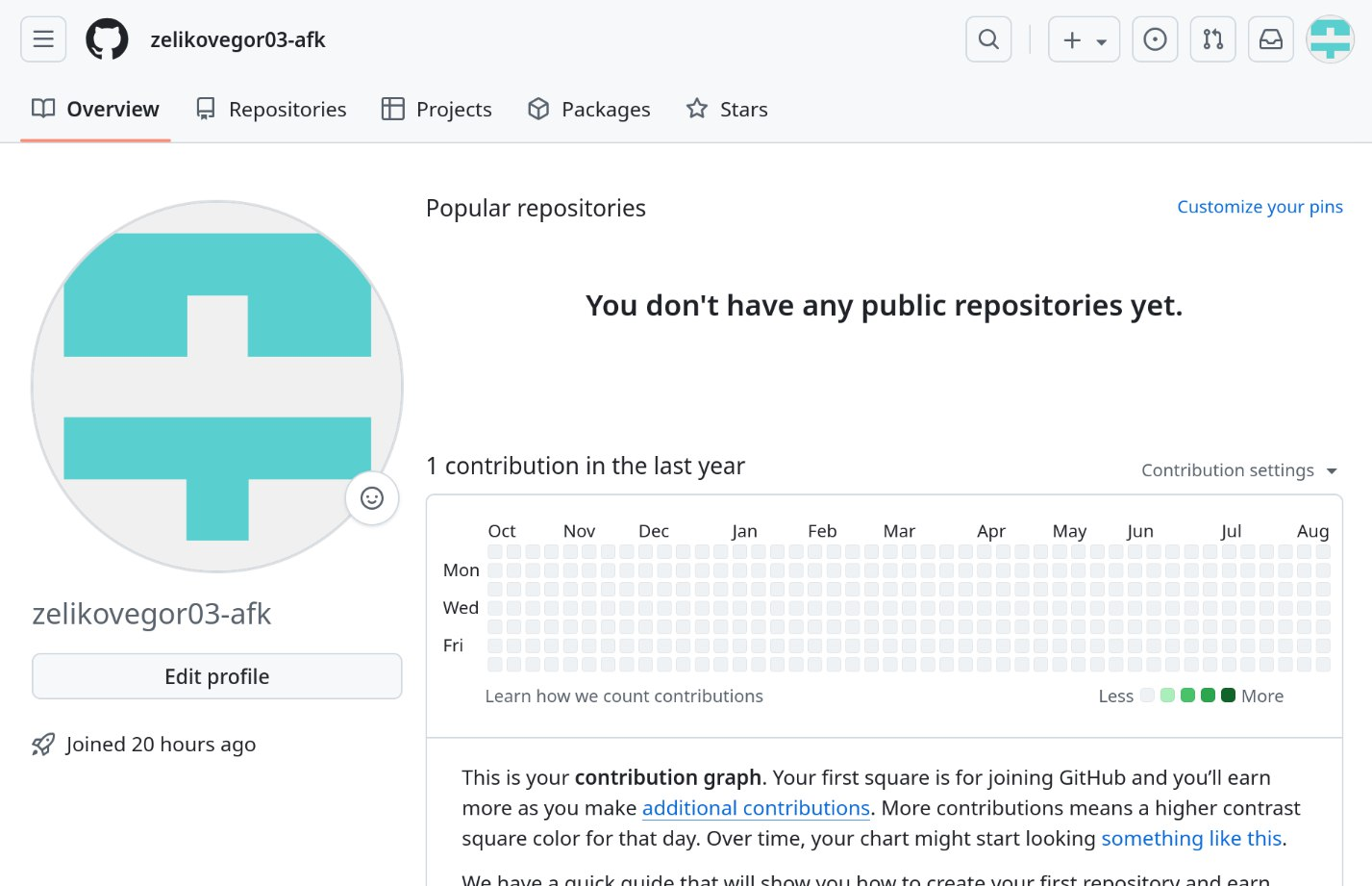
или

git push

# **3. Выполнение лабораторной работы**

## **3.1. Настройка github**

Создал учётную запись на сайте <https://github.com/> и заполнил основные данные:

[[1]](#footnote-10815)

## **3.2. Базовая настройка git**

Сначала сделал предварительную конфигурацию git. Открыл терминал и ввел следующие команды, указав своё имя и e-mail:

sapolyakovdk2n26@vbox:~$ git config --global user.name "<Поляков Сергей>"

sapolyakovdk2n26@vbox:~$ git config --global user.email "<zelikovegor03@gmail.com>"

Настроил utf-8 в выводе сообщений git:

sapolyakovdk2n26@vbox:~$ git config --global core.quotepath false

Задал имя начальной ветки (будем называть её master):

sapolyakovdk2n26@vbox:~$ git config --global init.defaultBranch master

Параметр autocrlf:

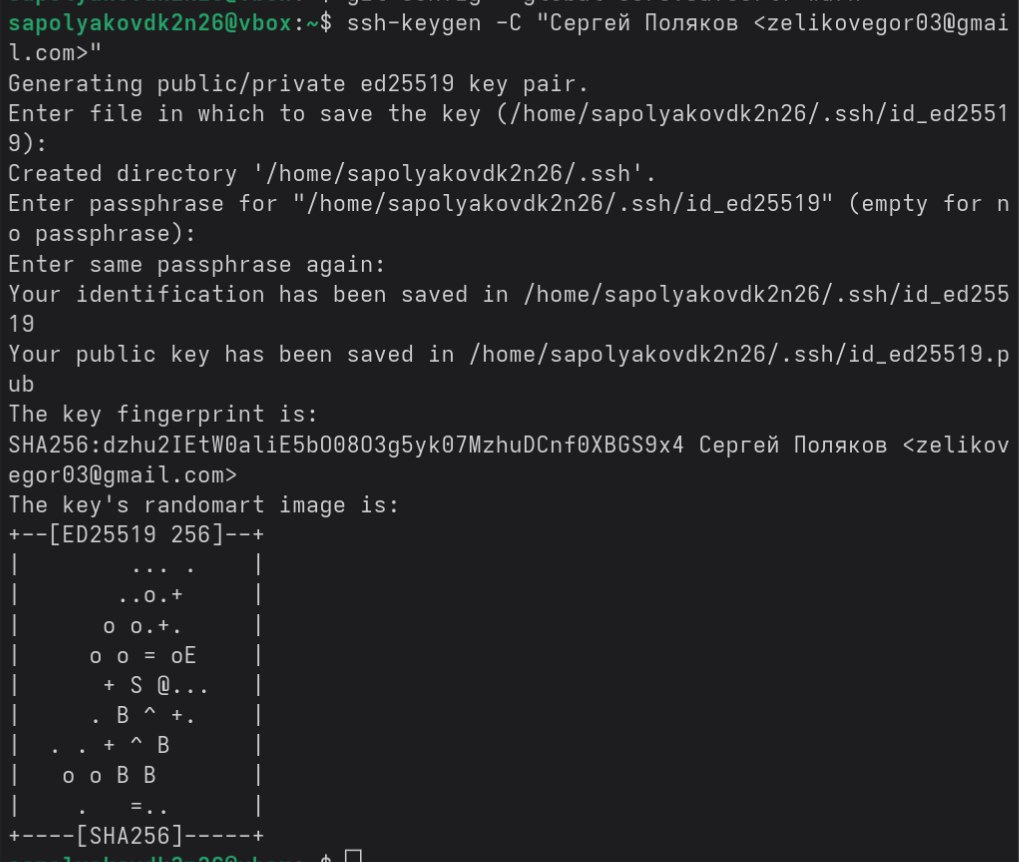
sapolyakovdk2n26@vbox:~$ git config --global core.autocrlf input

Параметр safecrlf:

sapolyakovdk2n26@vbox:~$ git config --global core.safecrlf warn

[[2]](#footnote-21327)

## **3.3. Создание SSH-ключа**

Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый): sapolyakovdk2n26@vbox:~$ ssh-keygen -C "Сергей Поляков [<zelikovegor03@gmail.com](mailto:<zelikovegor03@gmail.com)>[[3]](#footnote-6348)

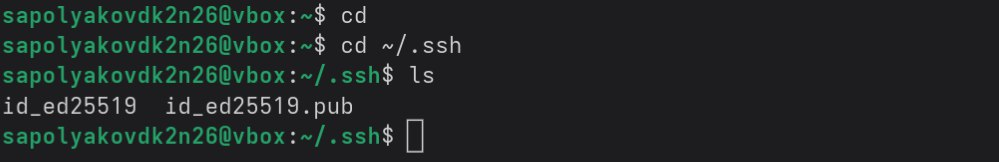
Ключи хранятся в каталоге

sapolyakovdk2n26@vbox:~$ cd

sapolyakovdk2n26@vbox:~$ cd ~/.ssh

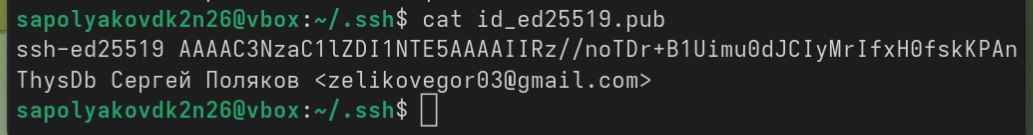
sapolyakovdk2n26@vbox:~/ .ssh$ ls

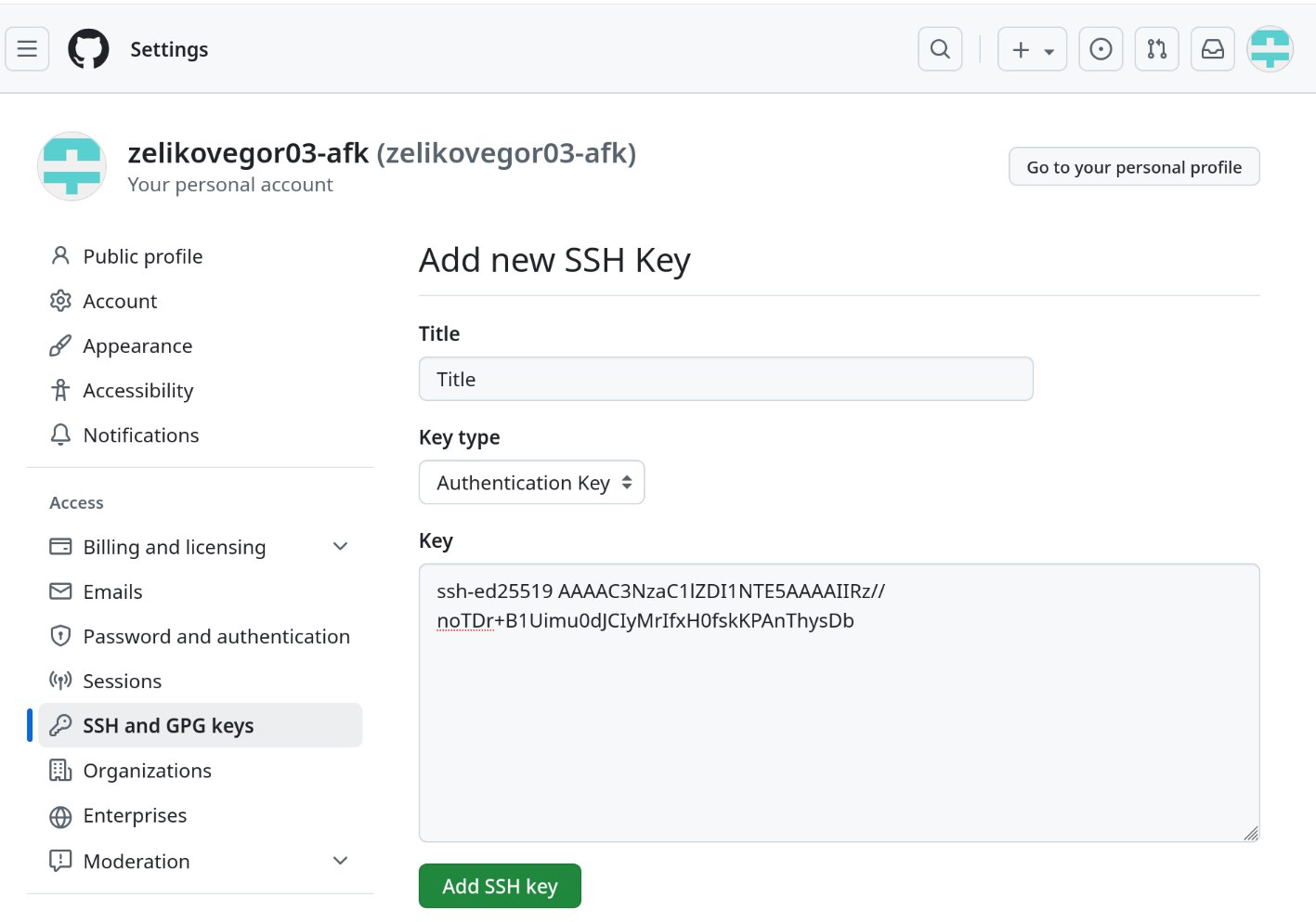
id\_ed25519 ; id\_ed25519.pub

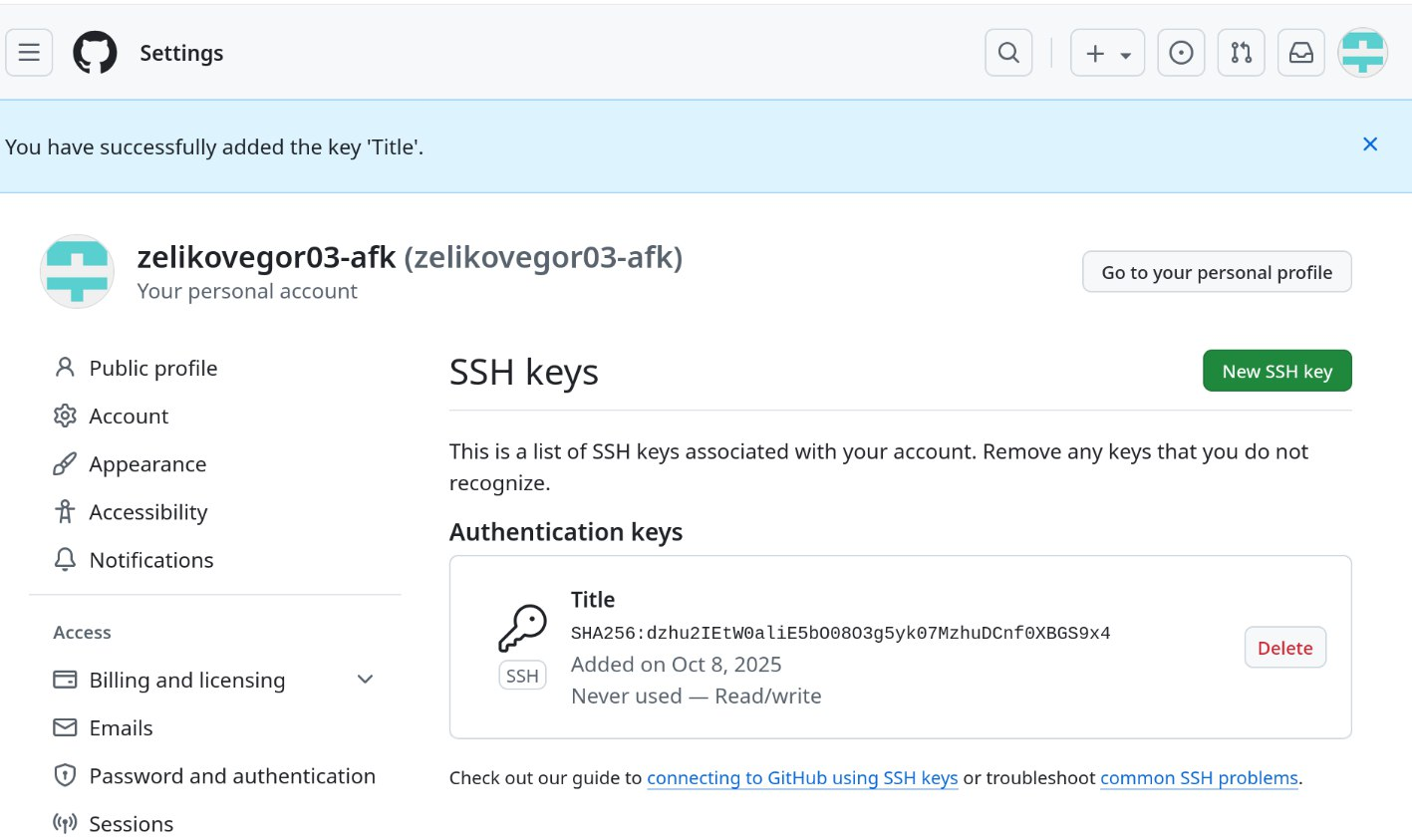
[[4]](#footnote-28863)

Далее необходимо загрузить сгенерированный открытый ключ. Для этого захожу на сайт <http://github.org/> под моей учётной записью и перехожу в меню Setting . После этого выбираю в боковом меню SSH and GPG keys и нажимаю кнопку New SSH key . Копирую из локальной консоли ключ в буфер обмена:

sapolyakovdk2n26@vbox:~/ .shh$ cat id\_ed25519.pub

[[5]](#footnote-10441)

Вставляю ключ в появившееся на сайте поле и указываю для ключа имя (Title). (Рисунок 6-7):[[6]](#footnote-254)

[[7]](#footnote-8847)

## **3.4. Создание рабочего пространства и репозитория курса на основе шаблонов**

При выполнении лабораторных работ следует придерживаться структуры рабочего про-странства. Рабочее пространство по предмету располагается в следующей иерархии:

~/work/study/

└── <учебный год>/

└── <название предмета>/

└── <код предмета>/

Например, для 2025–2026 учебного года и предмета «Архитектура компьютера» (код предмета arch-pc) структура каталогов примет следующий вид:

~/work/study/

└── 2025–2026/

└── Архитектура компьютера/

└── arch-pc/

└── labs/

└── lab01/

└── lab02/

└── lab03/

. . .

• Каталог для лабораторных работ имеет вид labs.

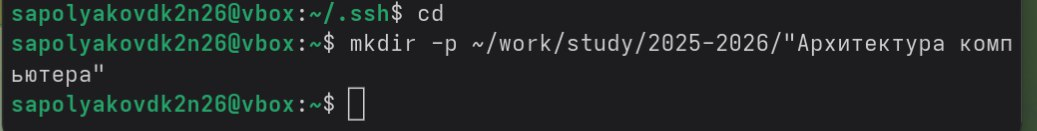
• Каталоги для лабораторных работ имеют вид lab<номер>, например: lab01, lab02 и т. д.

Название проекта на хостинге git имеет вид: study\_<учебный год>\_<код предмета> Например, для 2025–2026 учебного года и предмета «Архитектура компьютера» (код пред- мета arch-pc) название проекта примет следующий вид: study\_2025–2026\_arch-pc

Открыл терминал и создал каталог для предмета «Архитектура компьютера»:

sapolyakovdk2n26@vbox:~$ cd

sapolyakovdk2n26@vbox:~$ mkdir -p ~/work/study/2025-2026/"Архитектура компьютера"

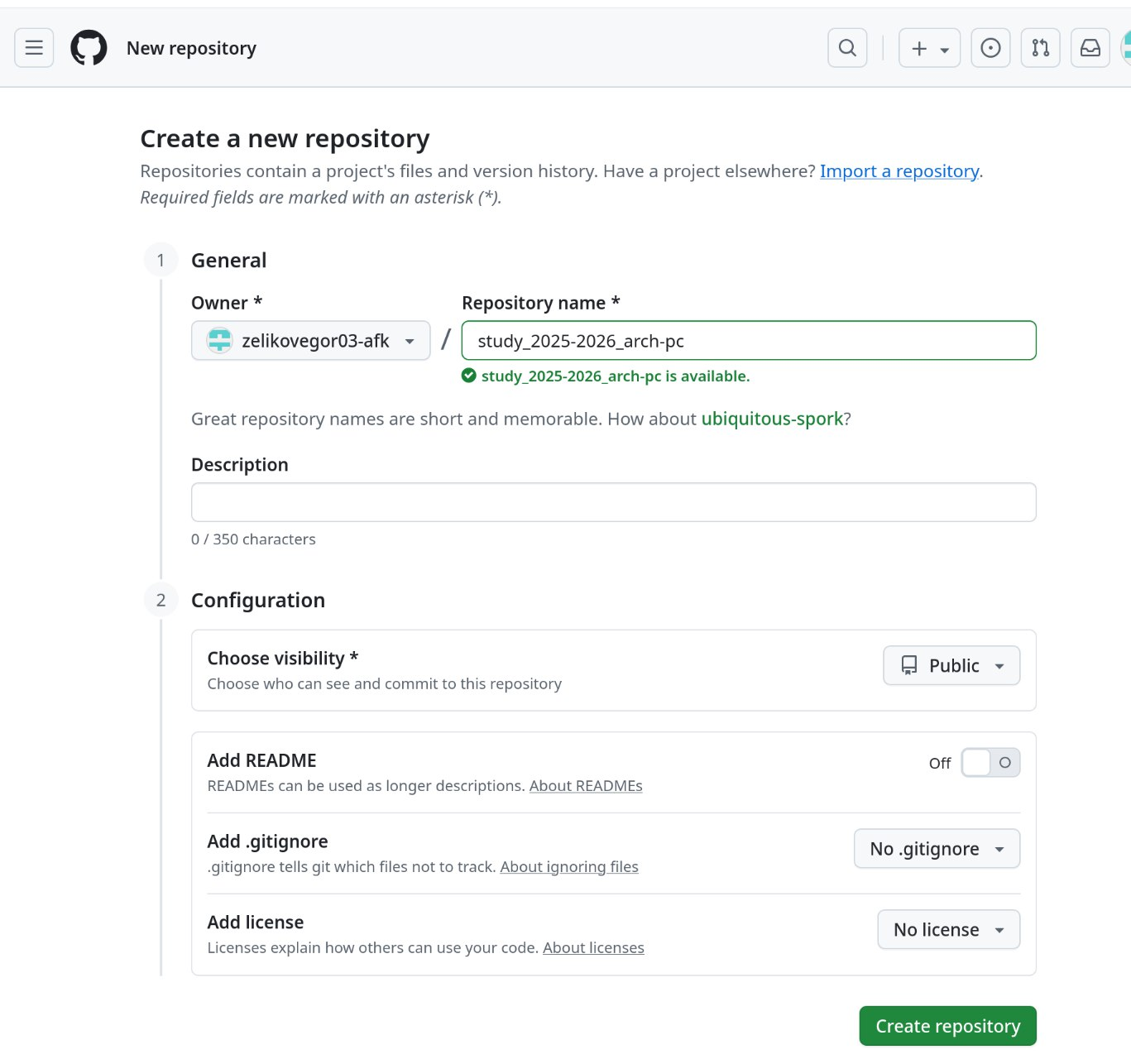
[[8]](#footnote-285)

## **3.5. Создание репозитория курса на основе шаблона**

Репозиторий на основе шаблона можно создать через web-интерфейс github. Перешел на станицу репозитория с шаблоном курса <https://github.com/yamadharma/course-directory-student-template>.

Далее зашёл в Use this template.

В открывшемся окне задал имя репозитория (Repository name) study\_2025–2026\_arh-pc и создал репозиторий (кнопка Create repository from template).

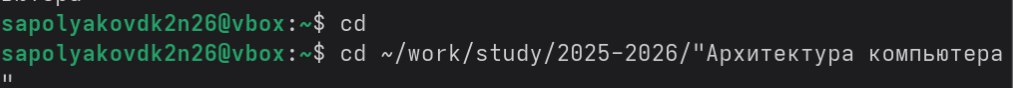
[[9]](#footnote-23062)

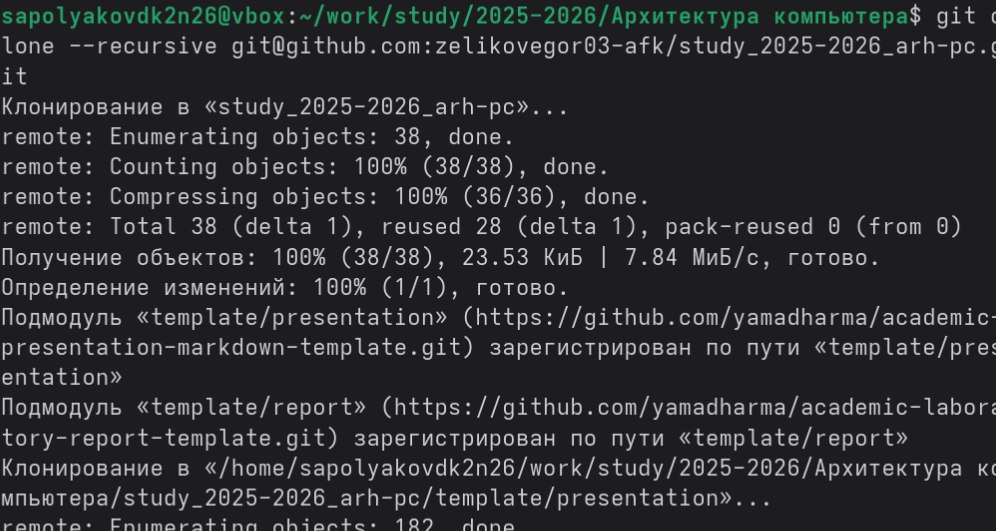
Зашёл в терминал и перешел в каталог курса, клонировал созданный репозиторий: (Рис 11-12)

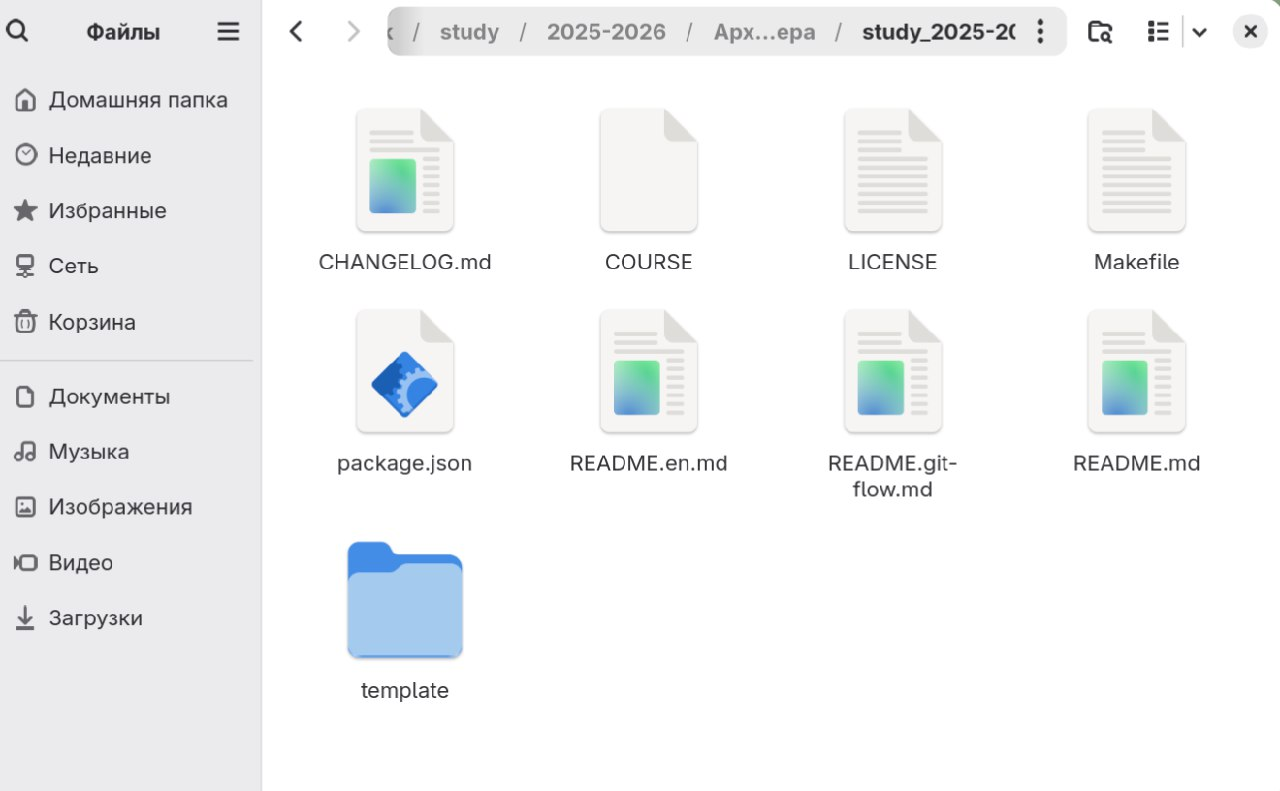
sapolyakovdk2n26@vbox:~$ cd

sapolyakovdk2n26@vbox:~$ cd ~/work/study/2025-2026/"Архитектура компьютера"

sapolyakovdk2n26@vbox:~/work/study/2025-2026/Архитектура компьютера$ git clone --recursive git@github.com:zelikovegor03-afk/study\_2025-2026\_arh-pc.git

[[10]](#footnote-29867)

[[11]](#footnote-30066)

[[12]](#footnote-562)

## **3.6. Настройка каталога курса**

Перешел в каталог курса: (Рис 13)

sapolyakovdk2n26@vbox:~/work/study/2025-2026/Архитектура компьютера$ cd

sapolyakovdk2n26@vbox:~$ cd ~/work/study/2025-2026/"Архитектура компьютера"/study\_2025-2026\_arh-pc

Создал необходимые каталоги: (Рис 13)

sapolyakovdk2n26@vbox:~/work/study/2025-2026/Архитектура компьютера/study\_2025-2026\_arh-pc$ echo arch-pc > COURSE

sapolyakovdk2n26@vbox:~/work/study/2025-2026/Архитектура компьютера/study\_2025-2026\_arh-pc$ make prepare

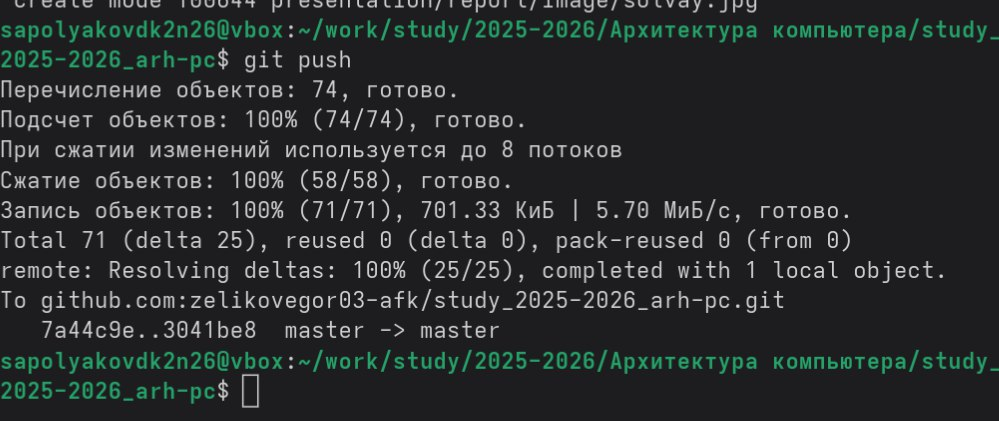
Отправил файлы на сервер: (Рис 14)

sapolyakovdk2n26@vbox:~/work/study/2025-2026/Архитектура компьютера/study\_2025-2026\_arh-pc$ git add .

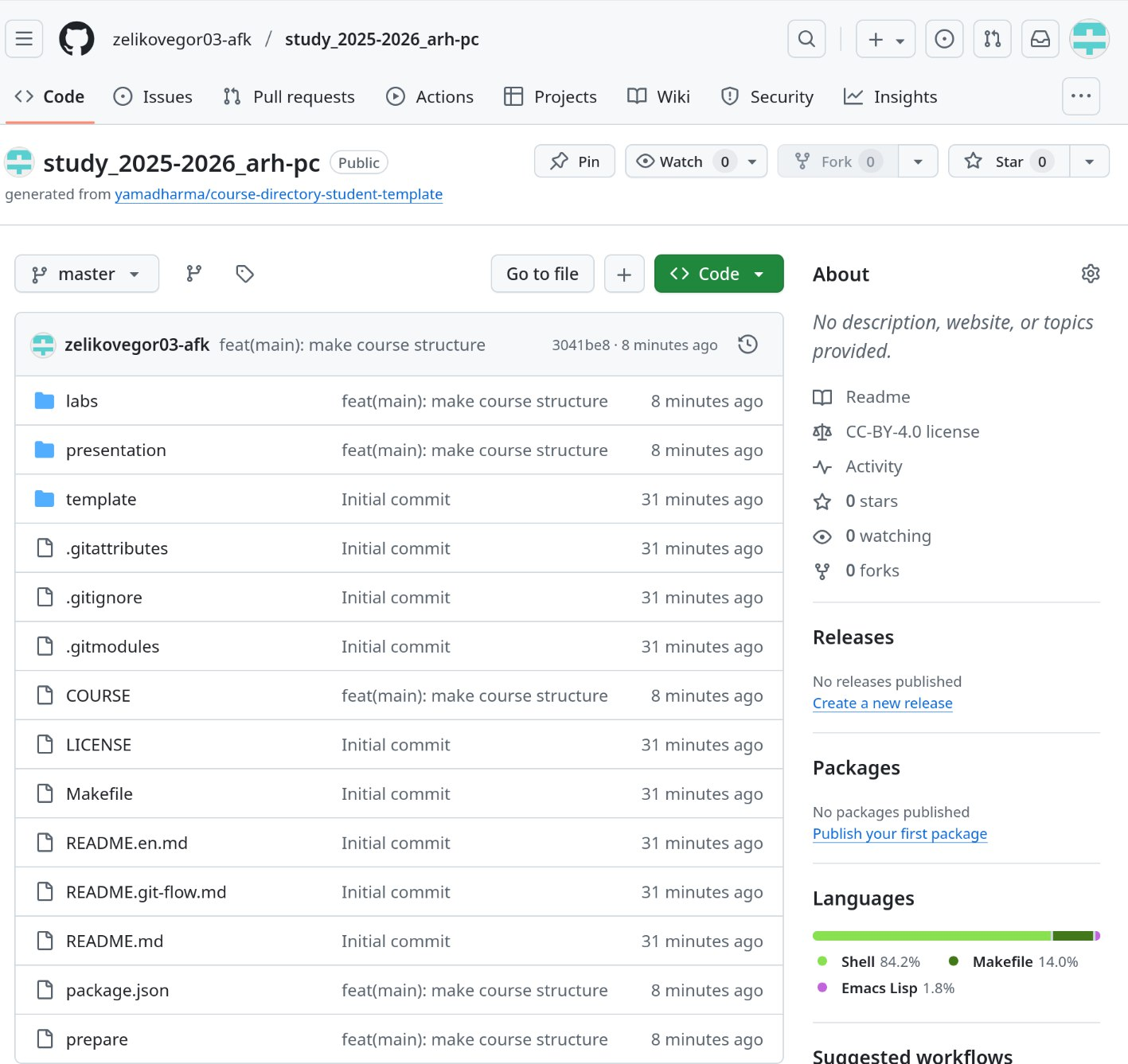
sapolyakovdk2n26@vbox:~/work/study/2025-2026/Архитектура компьютера/study\_2025-2026\_arh-pc$ git commit -am 'feat(main): make course structure'

sapolyakovdk2n26@vbox:~/work/study/2025-2026/Архитектура компьютера/study\_2025-2026\_arh-pc$ git push

[[13]](#footnote-5377)

[[14]](#footnote-8992)

Проверил правильность создания иерархии в локальном репозитории и на страницеgithub. (Рис 15)

[[15]](#footnote-5746)

# **4. Задание для самостоятельной работы**

1. Создал отчет по выполнению лабораторной работы в соответствующем каталоге

рабочего пространства (labs/lab02/report).

sapolyakovdk2n26@vbox:~/work/study/2025-2026/Архитектура компьютера/study\_2025-2026\_arh-pc$ cd

sapolyakovdk2n26@vbox:~$ mkdir -p labs/lab01/report labs/lab02/report

2. Скопировал отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства.

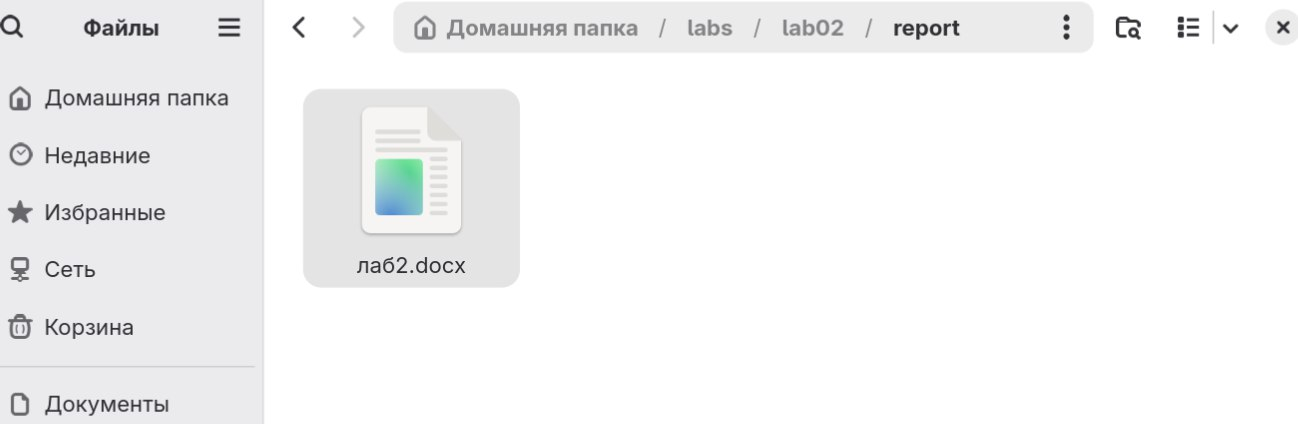
sapolyakovdk2n26@vbox:~$ cp ~/Загрузки/лаб2.docx labs/lab02/report/

sapolyakovdk2n26@vbox:~$ cp ~/Загрузки/лаб1.docx labs/lab01/report/

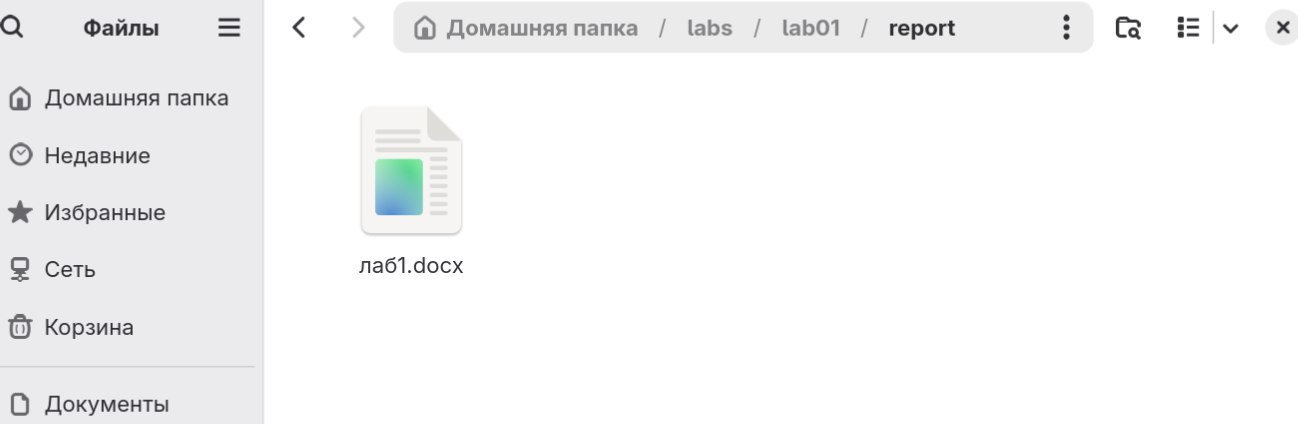
[[16]](#footnote-28400)

[[17]](#footnote-137)

См рис 16



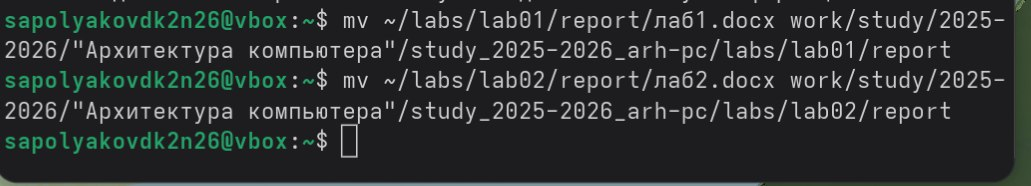
См рис 17



После этого переместил файлы в папку “Архитектура компьютера”: (Рис 18-20)

sapolyakovdk2n26@vbox:~$ mv ~/labs/lab01/report/лаб1.docx work/study/2025-2026/"Архитектура компьютера"/study\_2025-2026\_arh-pc/labs/lab01/report

sapolyakovdk2n26@vbox:~$ mv ~/labs/lab02/report/лаб2.docx work/study/2025-2026/"Архитектура компьютера"/study\_2025-2026\_arh-pc/labs/lab02/report

[[18]](#footnote-19661)

[[19]](#footnote-26985)

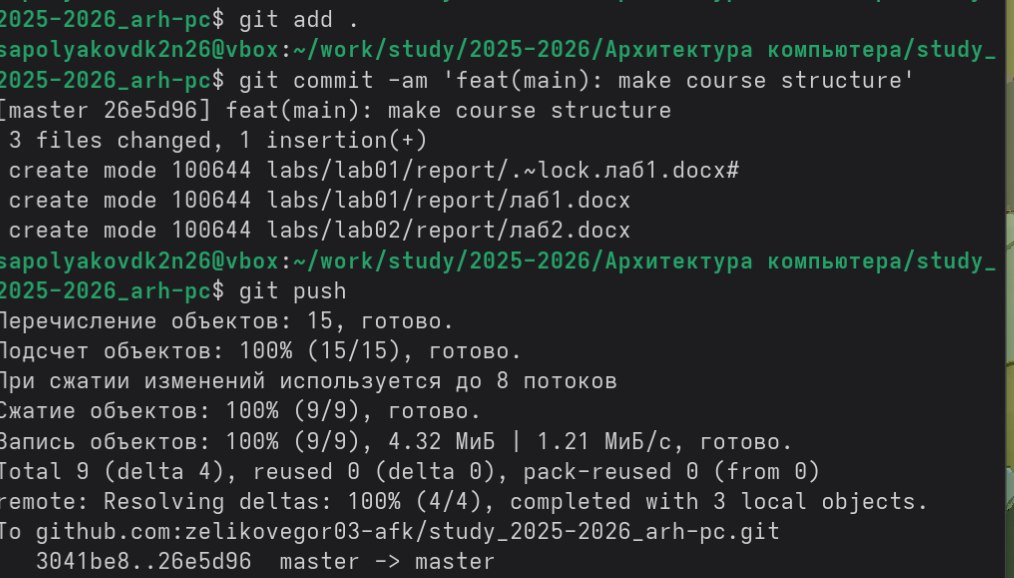
[[20]](#footnote-13256)

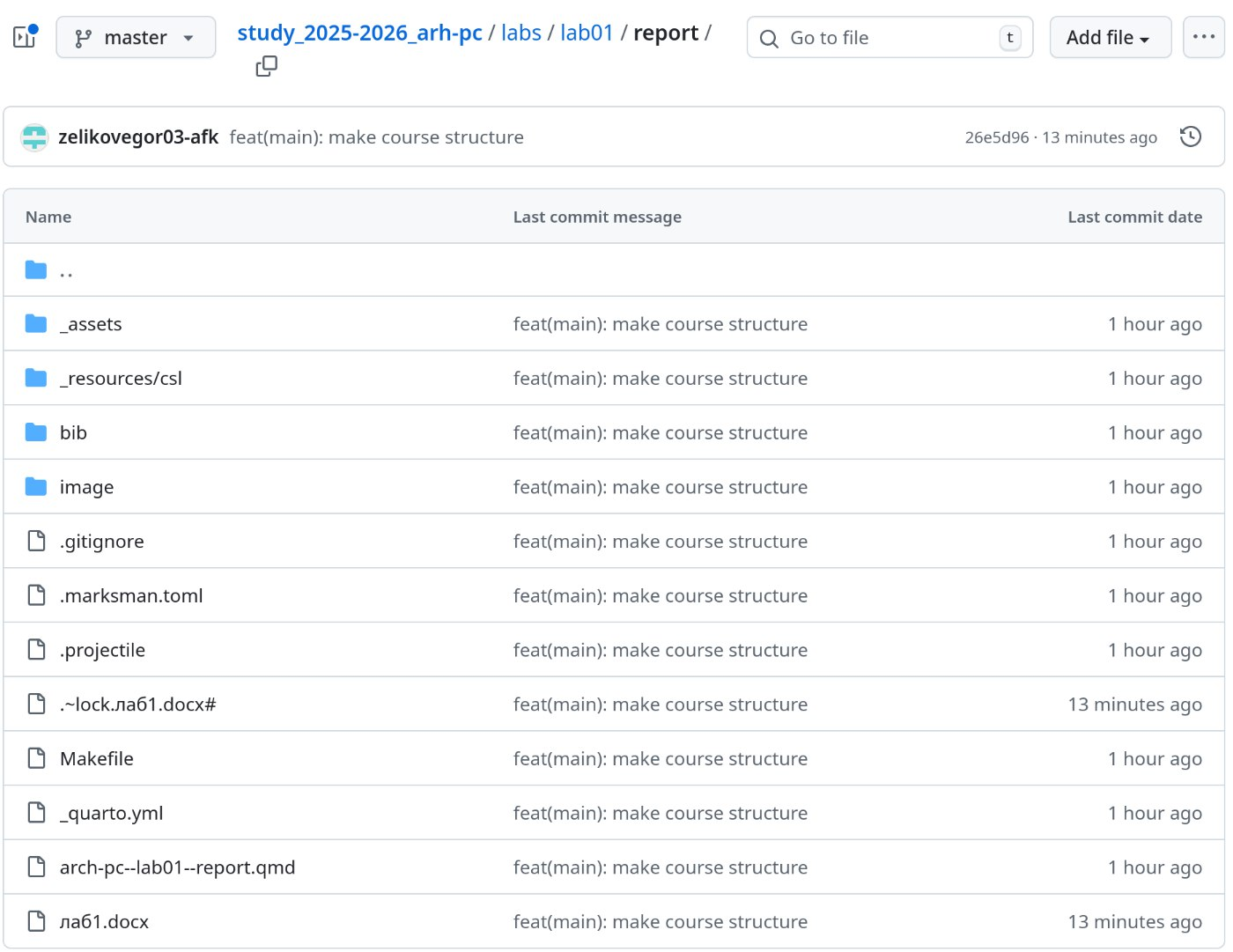
3. Загружу файлы на github.

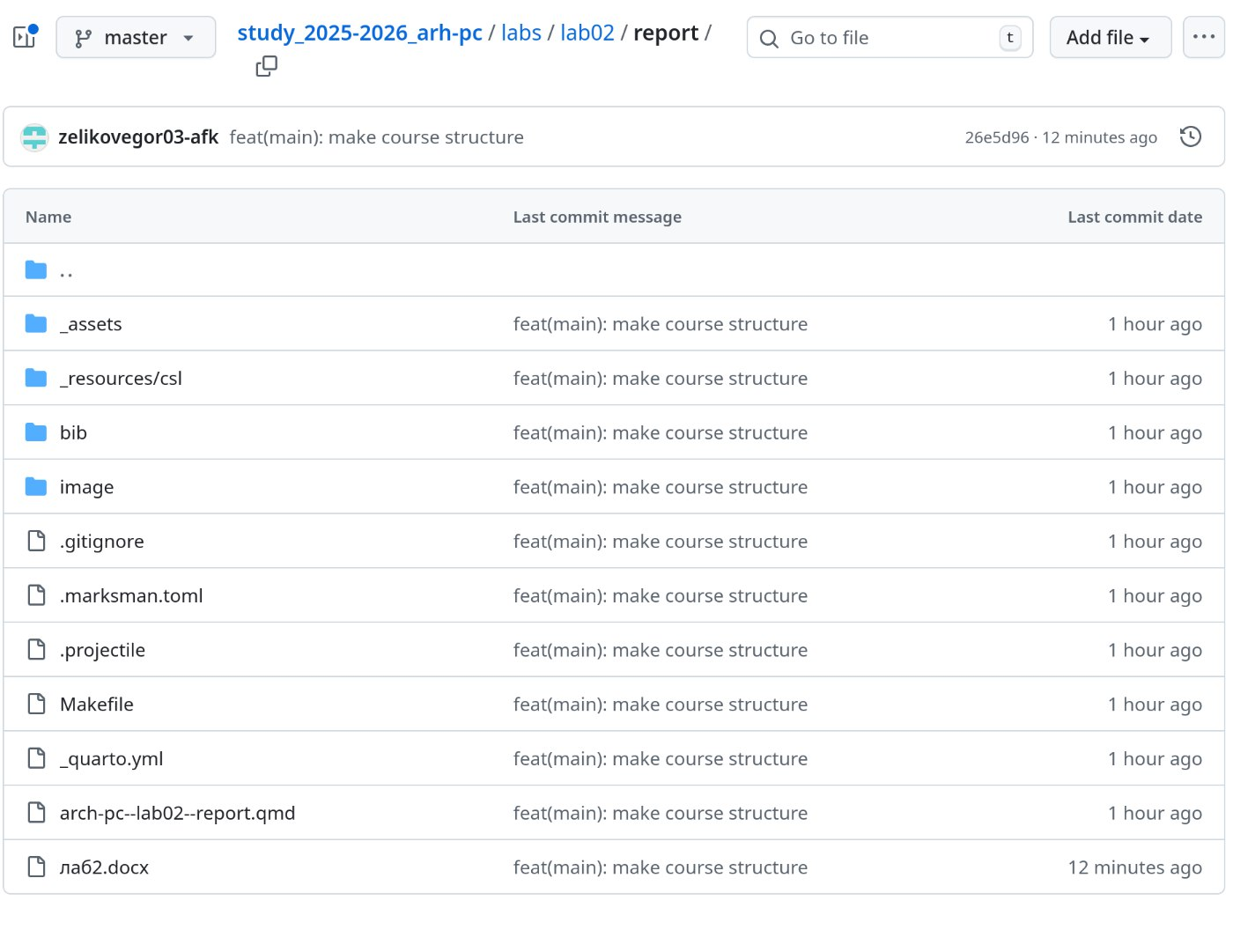
sapolyakovdk2n26@vbox:~/work/study/2025-2026/Архитектура компьютера/study\_2025-2026\_arh-pc$ git add .

sapolyakovdk2n26@vbox:~/work/study/2025-2026/Архитектура компьютера/study\_2025-2026\_arh-pc$ git commit -am 'feat(main): make course structure'

sapolyakovdk2n26@vbox:~/work/study/2025-2026/Архитектура компьютера/study\_2025-2026\_arh-pc$ git push

[[21]](#footnote-29937)

[[22]](#footnote-10851)[[23]](#footnote-12223)

[[24]](#footnote-6228)

# **5. Вывод**

В ходе работы были изучены принципы идеологии и применения средств контроля версий и приобретены практические навыки работы с системой Git, что обеспечивает готовность эффективно применять её в реальных проектах.

1. 3.1 1:создание профиля GitHub [↑](#footnote-ref-10815)
2. 3.2 1: Настройка Git [↑](#footnote-ref-21327)
3. 3.3 1: Создание ключа SSH [↑](#footnote-ref-6348)
4. 3.4 1: Место хранения ключа [↑](#footnote-ref-28863)
5. 3.5 1: Загрузка ключа [↑](#footnote-ref-10441)
6. 3.6 1: Cоздание ключа в Github [↑](#footnote-ref-254)
7. 3.7 1: Создание ключа в [↑](#footnote-ref-8847)
8. 3.8 1: Создание файлов ‘Архитектура компьютера’ [↑](#footnote-ref-285)
9. 3.9 1: Создание репозитория [↑](#footnote-ref-23062)
10. 3.10 1: Переход в подкатолог ”Архитектура компьютера” [↑](#footnote-ref-29867)
11. 3.11 1: Клонировал созданный репозиторий [↑](#footnote-ref-30066)
12. 3.12 1: Проверка файлов после созданных репозиториев [↑](#footnote-ref-562)
13. 3.13 1: Создание и отправка файлов [↑](#footnote-ref-5377)
14. 3.14 1: Отправка файлов [↑](#footnote-ref-8992)
15. 3.14 2: Проверка файлов в Github [↑](#footnote-ref-5746)
16. 4.1 1: Создание лаб работ и перенос [↑](#footnote-ref-28400)
17. 4.2 1: Создание и перенос лаб работ [↑](#footnote-ref-137)
18. 4.3 1: Перенос фалов в ”Архитектуру компьютера” [↑](#footnote-ref-19661)
19. 4.4 1: Проверка файлов [↑](#footnote-ref-26985)
20. 4.5 1: Проверка файлов [↑](#footnote-ref-13256)
21. 4.5 2: Загрузка файлов в GitHub [↑](#footnote-ref-29937)
22. 4.5 3: Проверка файлов на GitHub [↑](#footnote-ref-10851)
23. 4.5 3: Провекра [↑](#footnote-ref-12223)
24. 4.5 4: Проверка файлов на GitHub [↑](#footnote-ref-6228)