РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2. дисциплина: Архитектура компьютера

Студент: Кудинов М.С.

Группа: НКАбд-03-24

МОСКВА

2024 г.

Содержание

1 Цель работы

Целью данной работы является изучение идеологии и применение средств контроля версий, а также 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. 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. 1). Далее я заполнил основные данные учетной записи.

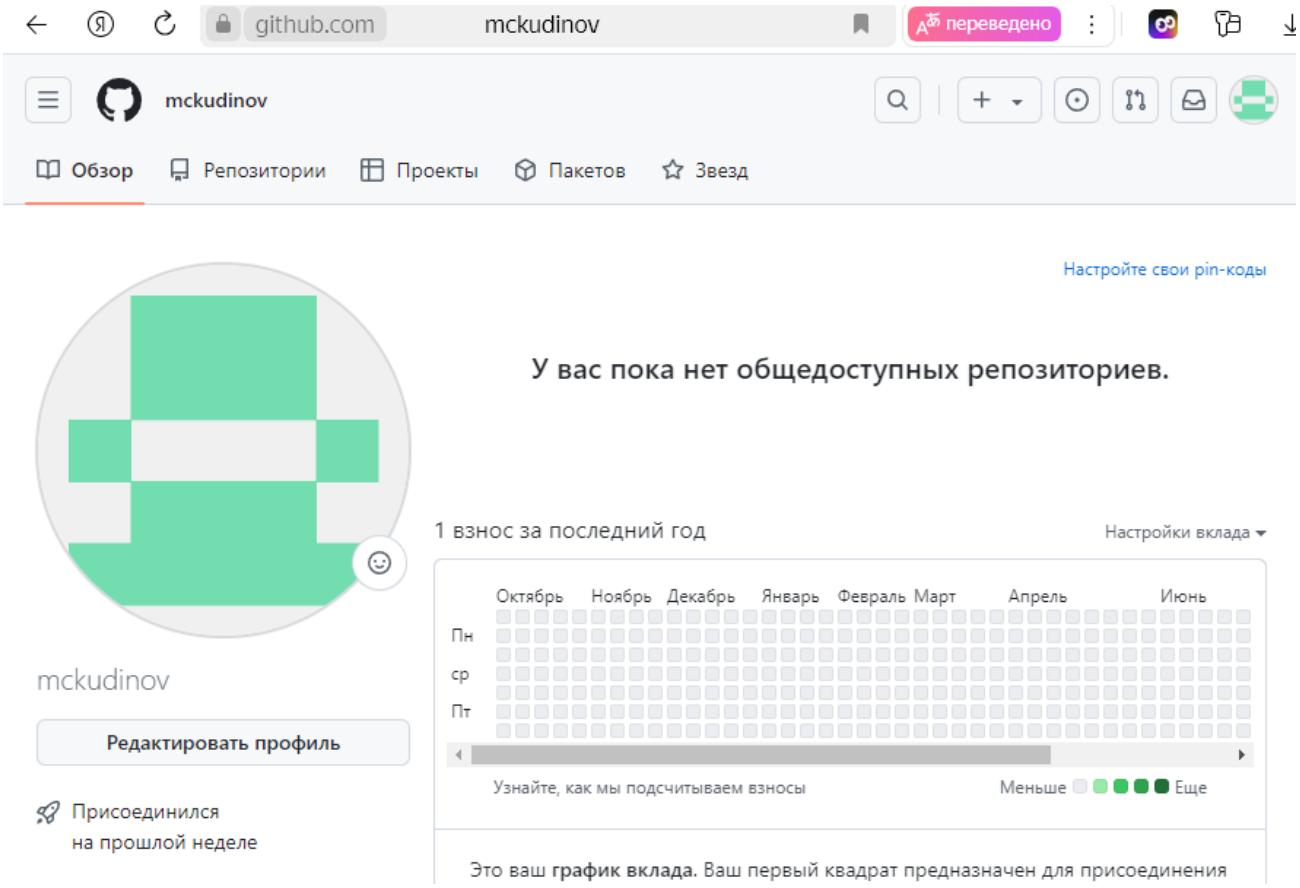

Рис.1

4.2 Базовая настройка Git

Открываю виртуальную машину, затем открываю терминал и делаю предварительную конфигурацию git. (рис. 2).

```
liveuser@mckudinov:~$ git config --global user.name "<mckudinov>"
liveuser@mckudinov:~$ git config --global user.email "<kudinov.maksim.20066@gmail.com>"
liveuser@mckudinov:~$ git config --global core.quotepath false
```

Рис.2

Задаю имя «master» для начальной ветки (рис. 3).

Задаю параметр autocrlf со значением input, так как я работаю в системе Linux, чтобы конвертировать CRLF в LF только при коммитах (рис. 3).

Задаю параметр safecrlf со значением warn, так Git будет проверять преобразование на обратимость (рис. 3). При значении warn Git только выведет предупреждение, но будет принимать необратимые конвертации.

```
liveuser@mckudinov:~$ git config --global init.defaultBranch master
liveuser@mckudinov:~$ git cinfig --global core.autocrlf input
git: 'cinfig' is not a git command. See 'git --help'.

The most similar command is
        config
liveuser@mckudinov:~$ git config --global core.autocrlf input
liveuser@mckudinov:~$ git config --global core.safecrlf warn
liveuser@mckudinov:~$
```

Рис.3

4.3 Создание SSH-ключа

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

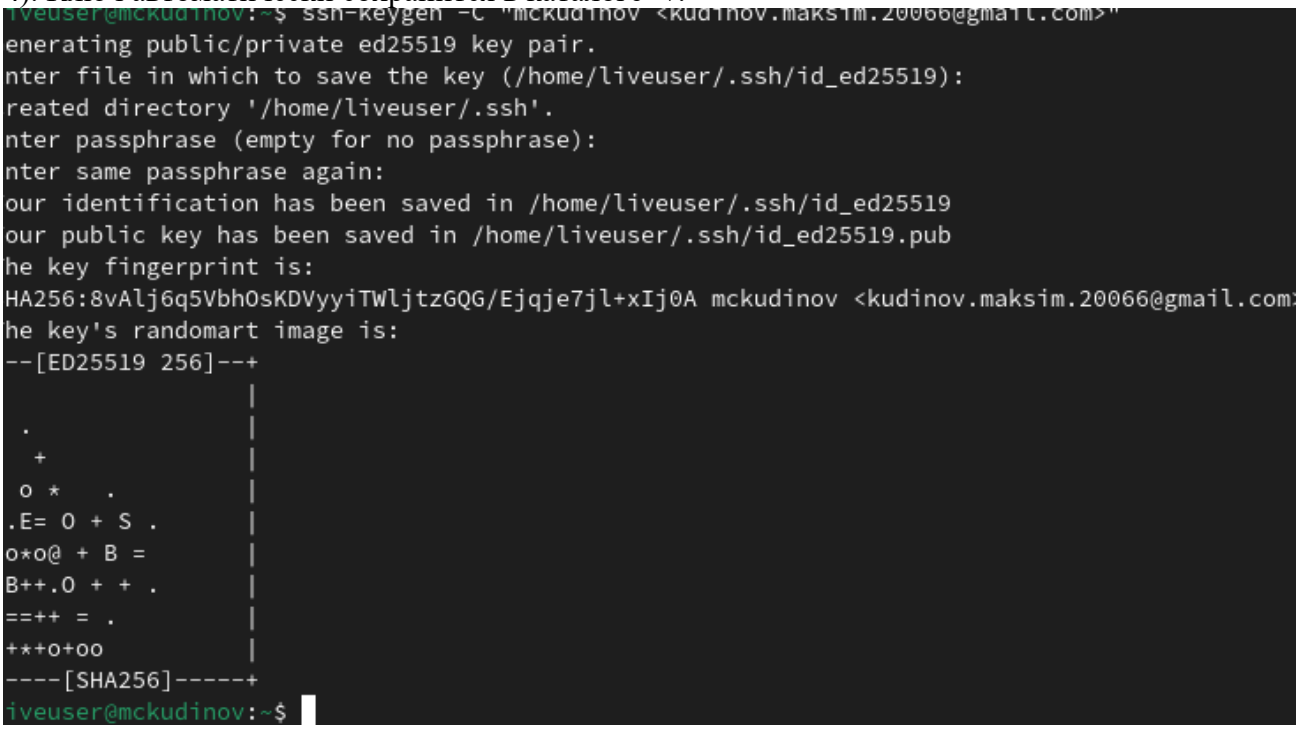

Рис. 4

Открываю браузер, захожу на сайт GitHub. Открываю свой профиль и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key» (рис. 15).

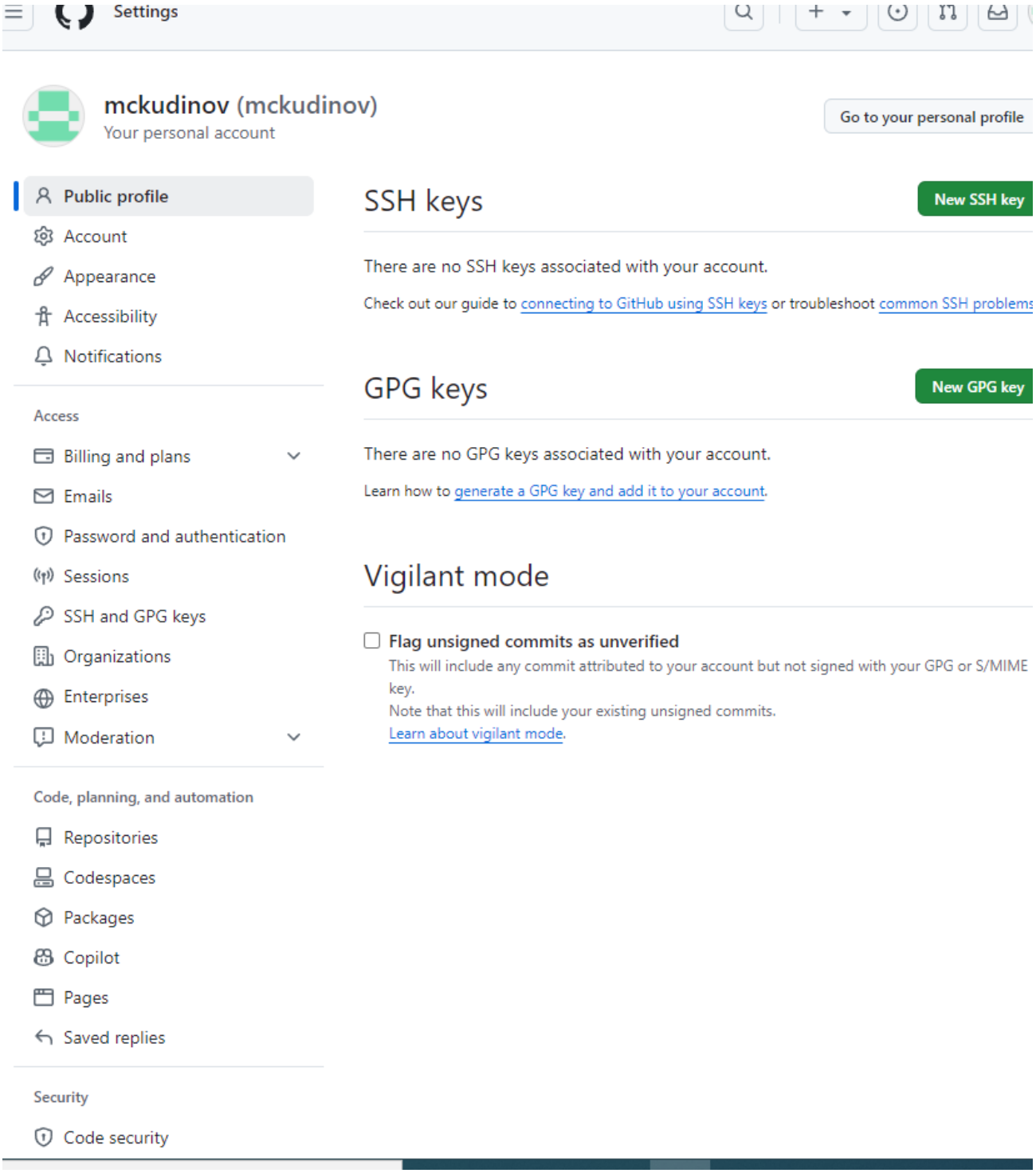
Settings
mckudinov (mckudinov)
Your personal account
Go to your personal profile
Public profile
Account
Appearance
Accessibility
Notifications
Access
Billing and plans
Emails
Password and authentication
Sessions
SSH and GPG keys
Organizations
Enterprises
Moderation
Code, planning, and automation
Repositories
Codespaces
Packages
Copilot
Pages
Saved replies
Security
Code security
SSH keys
New SSH key
There are no SSH keys associated with your account.
Check out our guide to connecting to GitHub using SSH keys or troubleshoot common SSH problems
GPG keys
New GPG key
There are no GPG keys associated with your account.
Learn how to generate a GPG key and add it to your account.
Vigilant mode
Flag unsigned commits as unverified
This will include any commit attributed to your account but not signed with your GPG or S/MIME key.
Note that this will include your existing unsigned commits.
Learn about vigilant mode.

Рис. 5

4.4 Создание рабочего пространства и репозитория курса на основе шаблона

Закрываю браузер, открываю терминал. Создаю директорию, рабочее пространство, с помощью утилиты mkdir, блягодаря 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. 13).

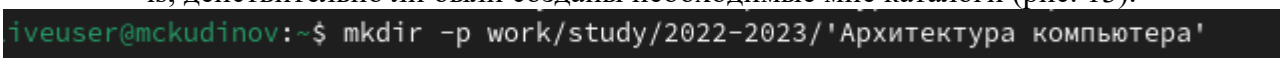

Рис. 6

4.5 Создание репозитория курса на основе шаблона

Через терминал перехожу в созданный каталог курса с помощью утилиты cd (рис. 7).

```
iveuser@mckudinov:~$ cd ~/work/study/2022-2023/'Архитектура компьютера'
```

Рис. 7

Клонирую созданный репозиторий с помощью команды git clone –recursive git@github.com:mckudinov/study_2023–2024_arh-pc.git arch-pc (рис. 8).

```
liveuser@mckudinov:~/work/study/2022-2023/Архитектура компьютера$ git clone --recursive git@github.com:mckudinov/stu
dy_2023-2024_arhpc.git
Cloning into 'study_2023-2024_arhpc'...
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)
Receiving objects: 100% (33/33), 18.82 KiB | 9.41 MiB/s, done.
Resolving deltas: 100% (1/1), done.
Submodule 'template/presentation' (https://github.com/yamadharma/academic-presentation-markdown-template.git) regist
ered for path 'template/presentation'
Submodule 'template/report' (https://github.com/yamadharma/academic-laboratory-report-template.git) registered for p
ath 'template/report'
Cloning into '/home/liveuser/work/study/2022-2023/Архитектура компьютера/study_2023-2024_arhpc/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)
Receiving objects: 100% (111/111), 102.17 KiB | 711.00 KiB/s, done.
Resolving deltas: 100% (42/42), done.
Cloning into '/home/liveuser/work/study/2022-2023/Архитектура компьютера/study_2023-2024_arhpc/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)
Receiving objects: 100% (142/142), 341.09 KiB | 1.81 MiB/s, done.
Resolving deltas: 100% (60/60), done.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5086c9c72a02bd2fca1d4a6'
Submodule path 'template/report': checked out 'c26e22effe7b3e0495707d82ef561ab185f5c748'
liveuser@mckudinov:~/work/study/2022-2023/Архитектура компьютера$
```

Рис. 8

Копирую ссылку для клонирования на странице созданного репозитория, сначала перейдя в окно «code», далее выбрав в окне вкладку «SSH» (рис. 19).

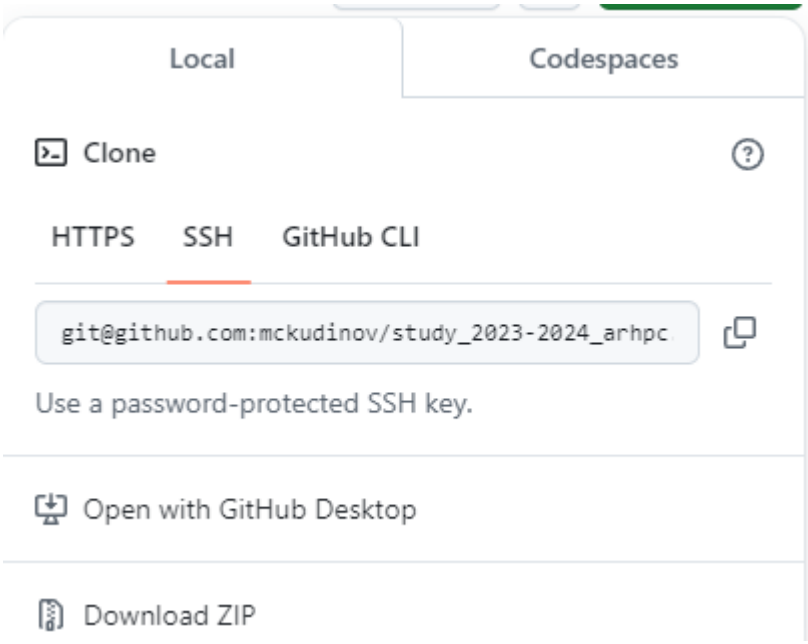

Рис. 9

4.6 Настройка каталога курса

Перехожу в каталог arch-pc с помощью утилиты cd (рис. 10).

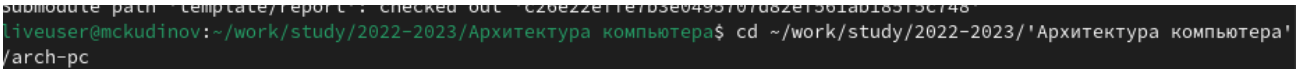

Рис. 10

Удаляю лишние файлы с помощью утилиты rm (рис. 11).

Рис.11

Создаю необходимые каталоги (рис. 12).

Отправляю созданные каталоги с 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. 12).

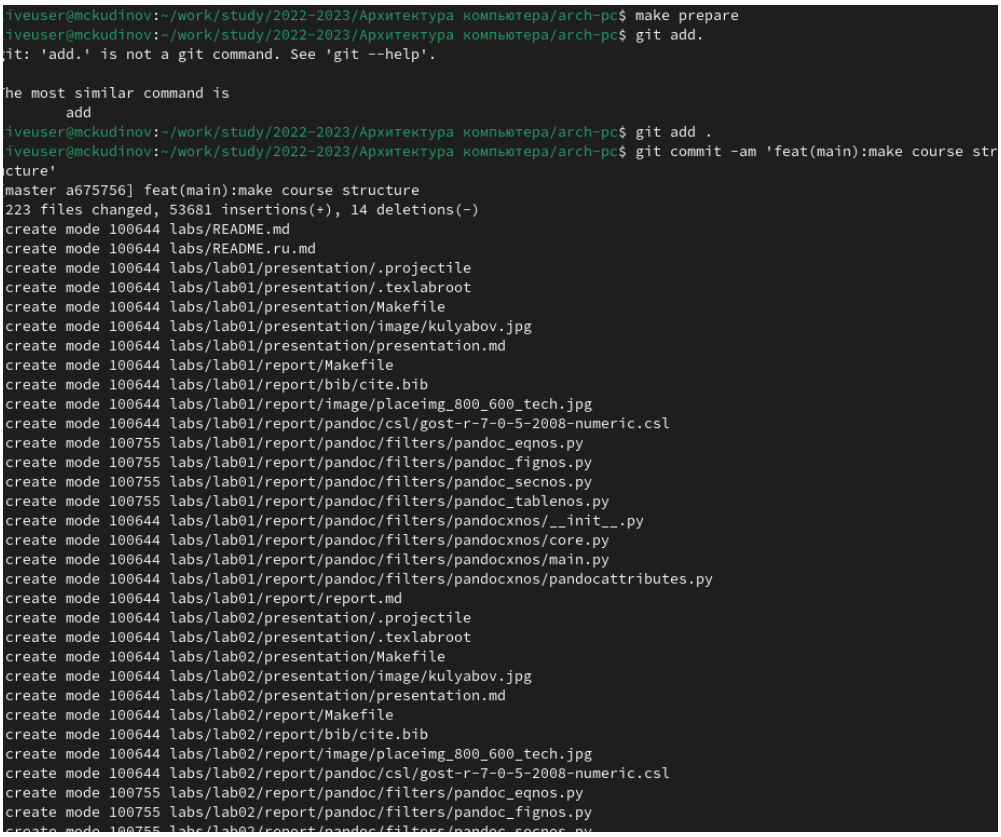

Рис. 12

Загружаю все в git(рис. 13).

```
liveuser@mckudinov:~/work/study/2022-2023/Архитектура компьютера/arch-pc$ git push
Enumerating objects: 37, done.
Counting objects: 100% (37/37), done.
Delta compression using up to 3 threads
Compressing objects: 100% (29/29), done.
Writing objects: 100% (35/35), 341.40 KiB | 3.25 MiB/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:mckudinov/study_2023-2024_arhpc.git
   4f69a03..a675756  master -> master
liveuser@mckudinov:~/work/study/2022-2023/Архитектура компьютера/arch-pc$
```

Рис. 13

Проверяю правильность выполнения работы сначала на самом сайте GitHub (рис.14).

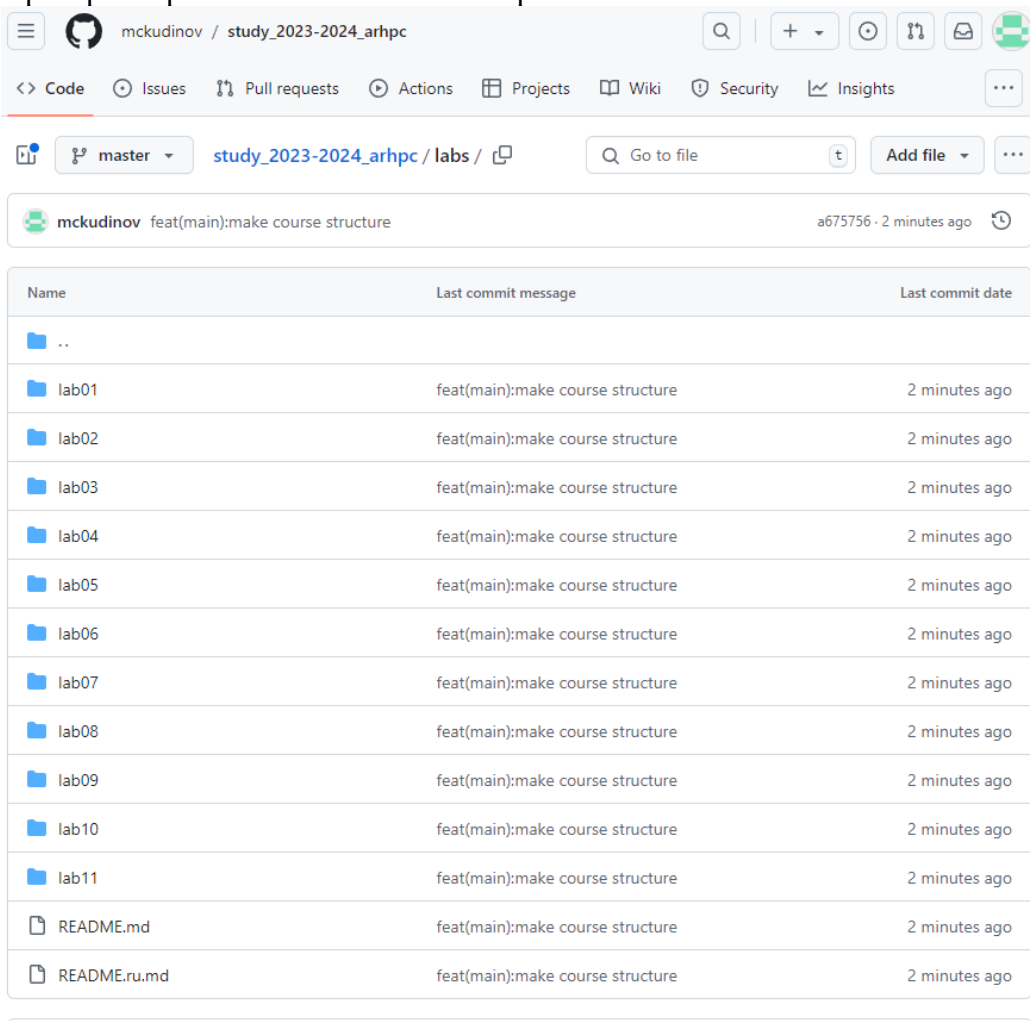

Рис. 14

4.7 Выполнение заданий для самостоятельной работы

Копирую лабораторную работу, используя ‘cp’(рис.15).

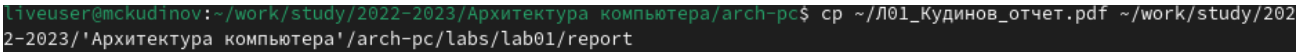

Рис.15

Захожу в “report” с помощью ‘cd’(рис.16).

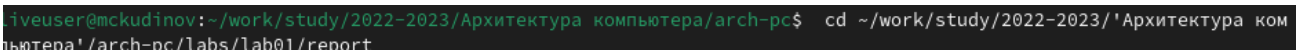

Рис. 16

Для отслеживания файлов использую “git add .”, после чего осуществляю перенос с помощью ‘git commit -am’ (рис.17). Остается отправить все в git, используя ‘git push’, но к сожалению мне выдало ошибку, которую я так и не смог решить.

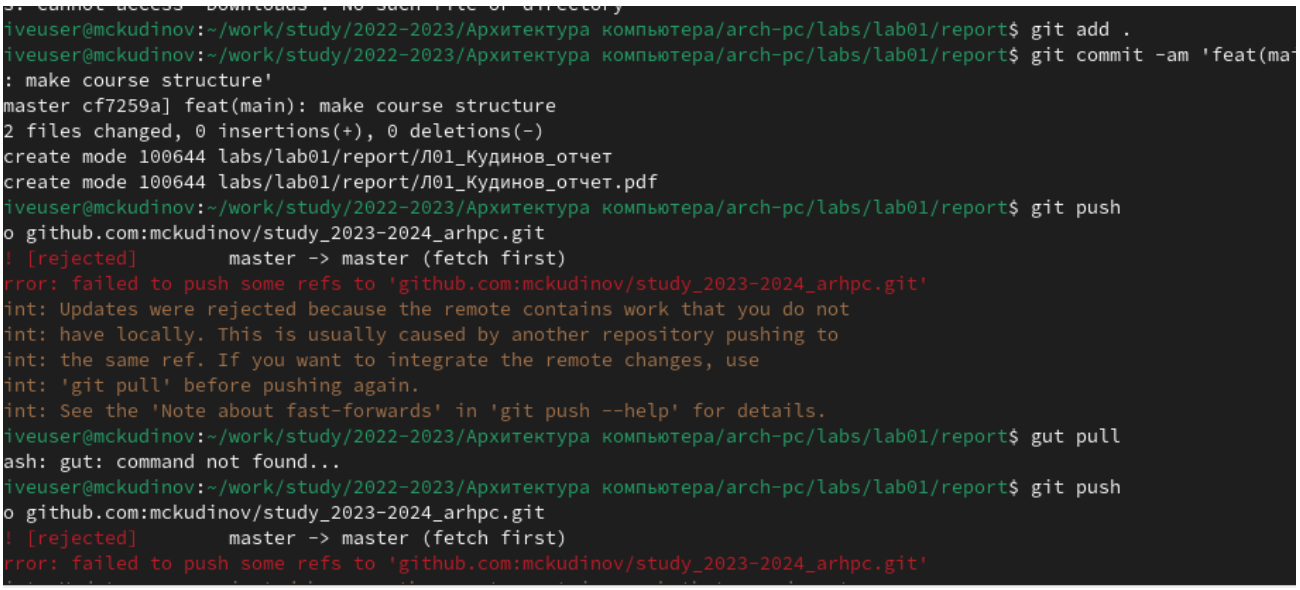

Рис. 17

5 Выводы

При выполнении данной лабораторной работы я изучил идеологию и применение средств контроля версий, а также приобрел практические навыки по работе с системой git.

6 Список литературы

1. Архитектура ЭВМ
2. Git - gitattributes Документация