**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: Кармазян Роберт Мушегович

Группа: НБИбд-01-23

**МОСКВА** 2023 г.

**Содержание**

1. Цель работы

2. Задание

3. Теоретическое введение

4. Выполнение лабораторной работы

4.1 Настройка GitHub

4.2 Базовая настройка Git

4.3 Создание SSH-ключа

4.4 Создание рабочего пространства и репозитория курса на основе шаблона

4.5 Создание репозитория курса на основе шаблона

4.6 Настройка каталога курса

4.7 Выполнение заданий для самостоятельной работы

5. Выводы

1. **Цель работы**

Целью работы является изучить идеологию и применение средств контроля версий. 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 работ

1. **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. 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. 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.

1. **Выполнение лабораторной работы**

**4.1 Настройка GitHub**

Создаю учётную запись на сайте <https://github>.com/ и заполните основные данные.

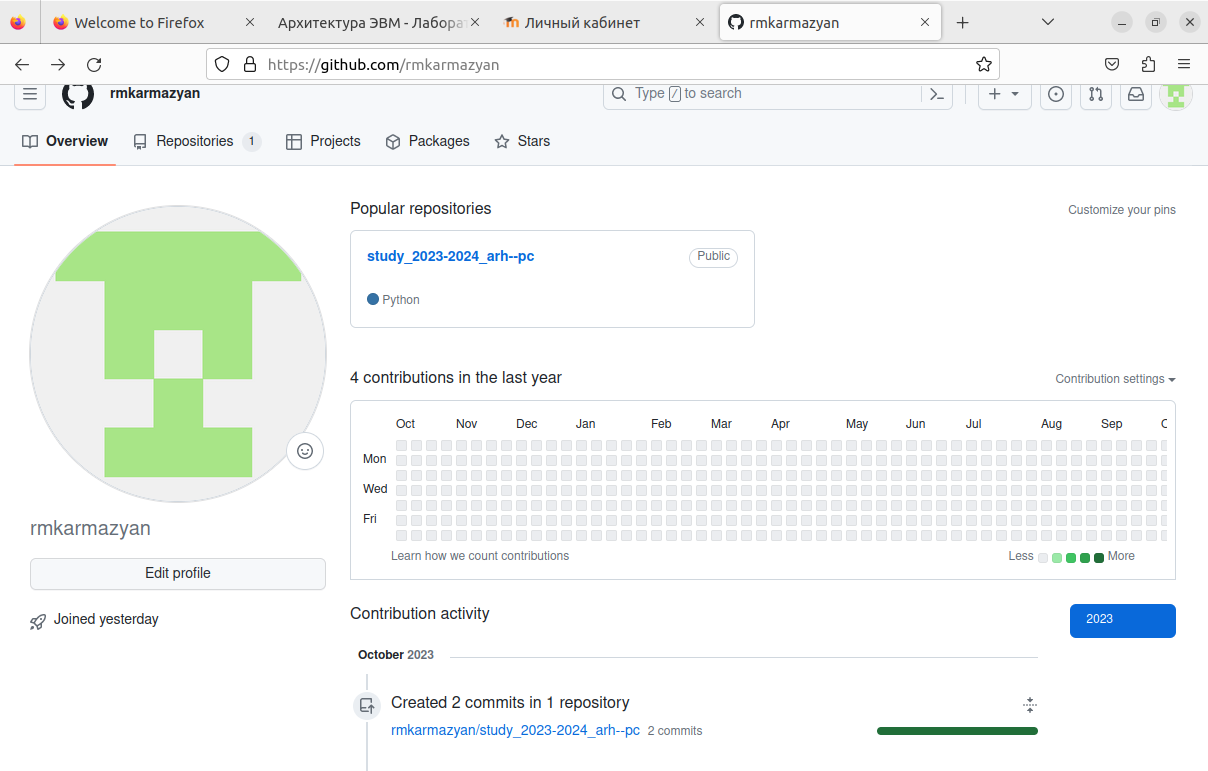


Рис. 4.1. Созданный аккаунт

**4.2. Базовая настройка Git**

Открыл терминал и ввел следующие команды, указав своё имя и email, далее настроил utf-8 в выводе сообщений git. Задал имя начальной ветки (будем называть её master). Также задал параметр autocrlf и параметру safecrlf.

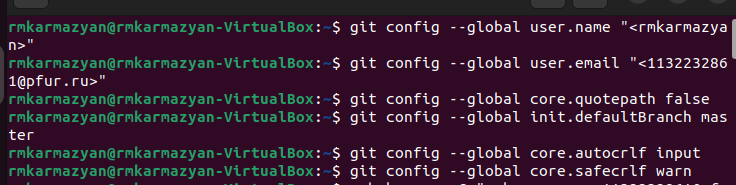
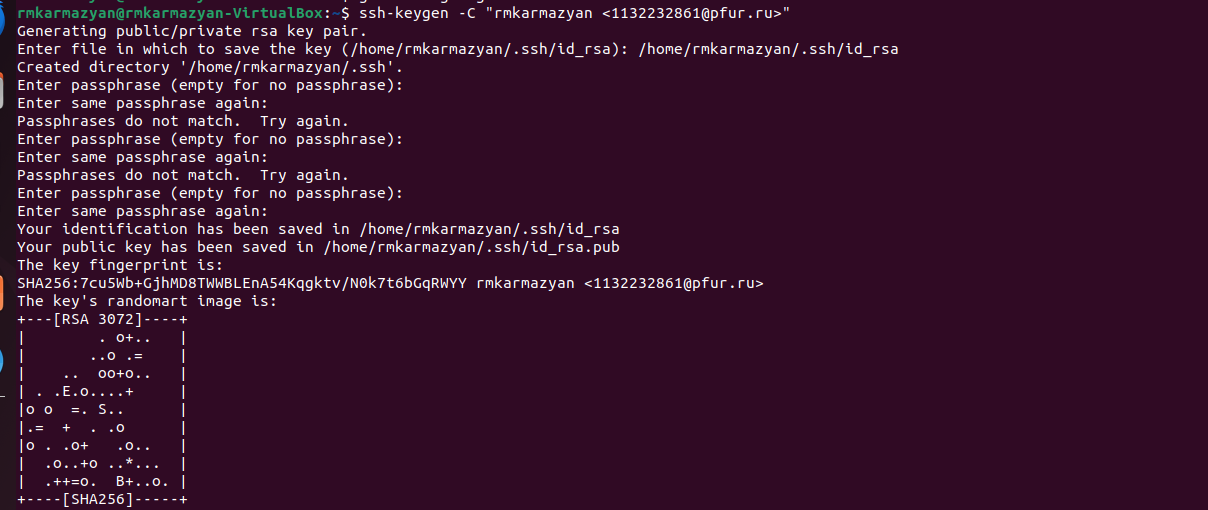


Рис 4.2. Команды базовой настройки git

**4.3. Создание SSH-ключа**

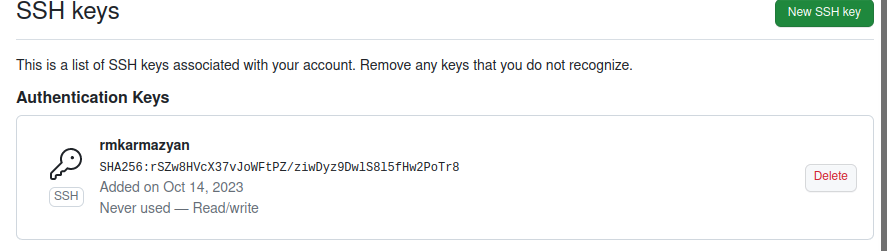
Для последующей идентификации пользователя на сервере репозиториев необходимо сгенерировать пару ключей (приватный и открытый). Для этого есть команда ssh-keygen -C "Имя Фамилия ". Ключи сохраняться в каталоге ~/.ssh/.

Рис. 4.3. Создание SSH ключа



Далее я скопировал ключ из файла KYES (также можно было использовать xclip) и вставил ключ в появившееся на сайте поле и указал для ключа имя.

Рис. 4.4. Созданный SSH ключ и добавленный на сайт github



**4.4. Создание рабочего пространства**

Открыл терминал и создайте каталог для предмета «Архитектура компьютера»



Рис. 4.5. Создание рабочего пространства

**4.5. Создание репозитория курса**

Репозиторий на основе шаблона можно создать через web-интерфейс github. Перешёл на станицу репозитория с шаблоном курса https://github.com/yamadharma/course-directory-student-template. Далее выбрал Use this template. В открывшемся окне задайте имя репозитория (Repository name) study\_2023–2024\_arhpc и создайте репозиторий (кнопка Create repository from template).

Рис. 4.6. Создание собственного репозитория

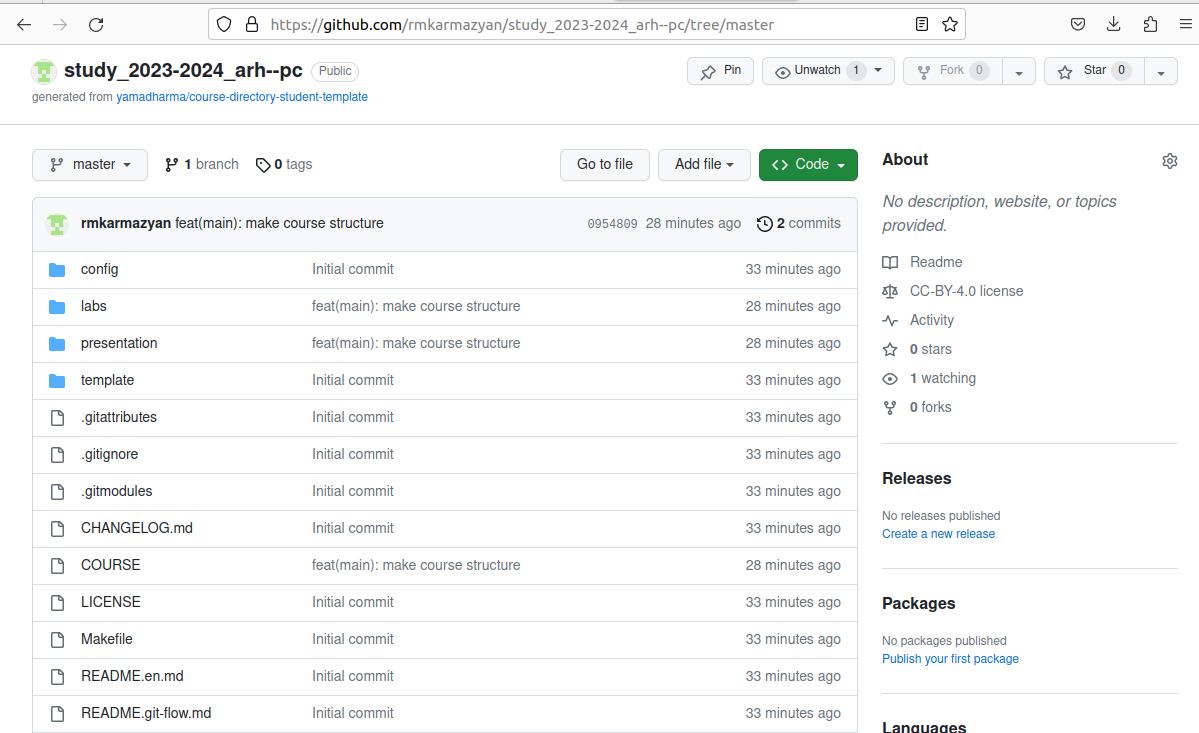


Рис. 4.7. Созданный репозиторий

Открыл терминал и перейдите в каталог курса

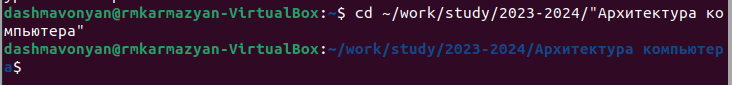


Рис. 4.8. Перемещение по директориям

Клонировал созданный репозиторий (Ссылку для клонирования можно скопировать на странице созданного репозитория Code -> SSH)

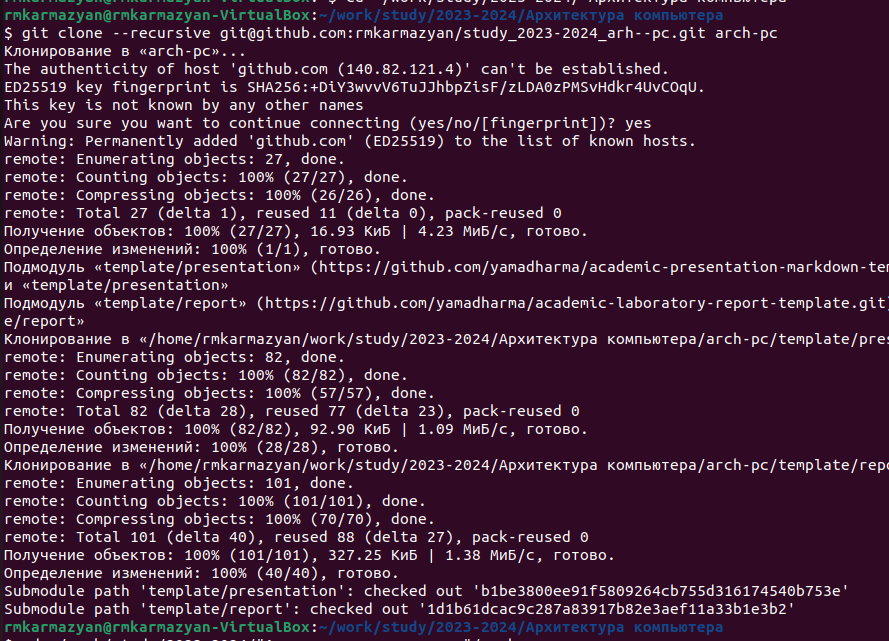
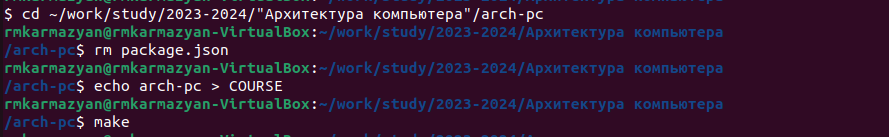


Рис. 4.9. Клонирование репозитория

**4.6. Настройка каталога курса**

Перешёл в каталог курса, удалил все лишние файлы и создал необходимые каталоги

Рис. 4.10. Внутренние настройки каталога курса 

Далее я отправляю файлы на сервер

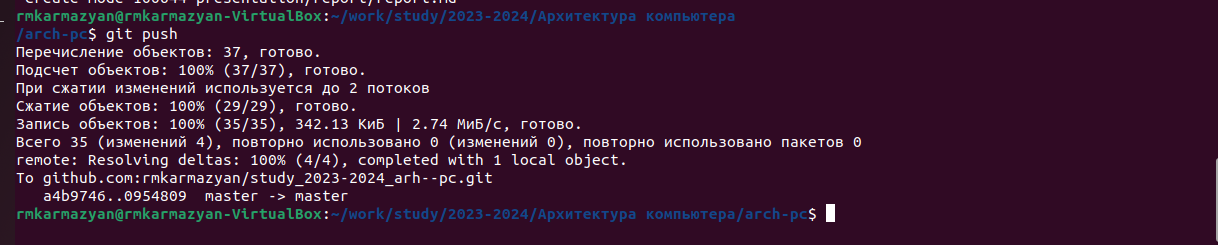
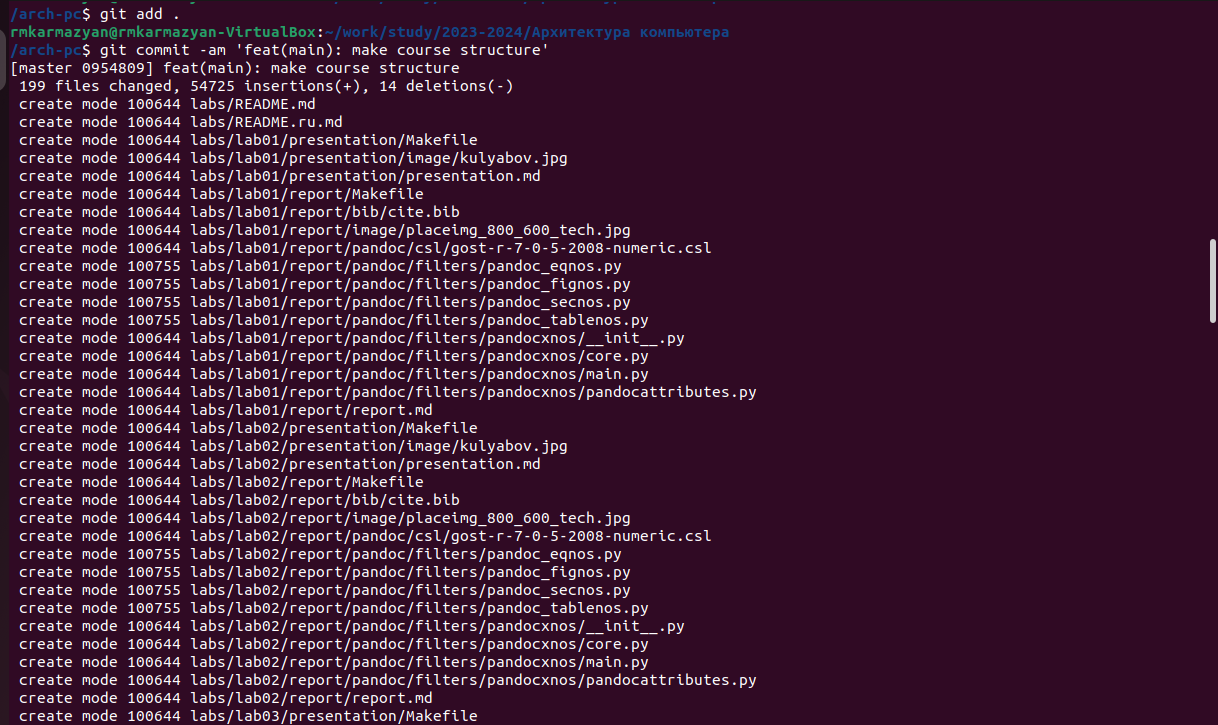


Рис. 4.11. Процесс отправления файлов на сервер

Проверяю правильность создания иерархии рабочего пространства в локальном репозитории и на странице github.

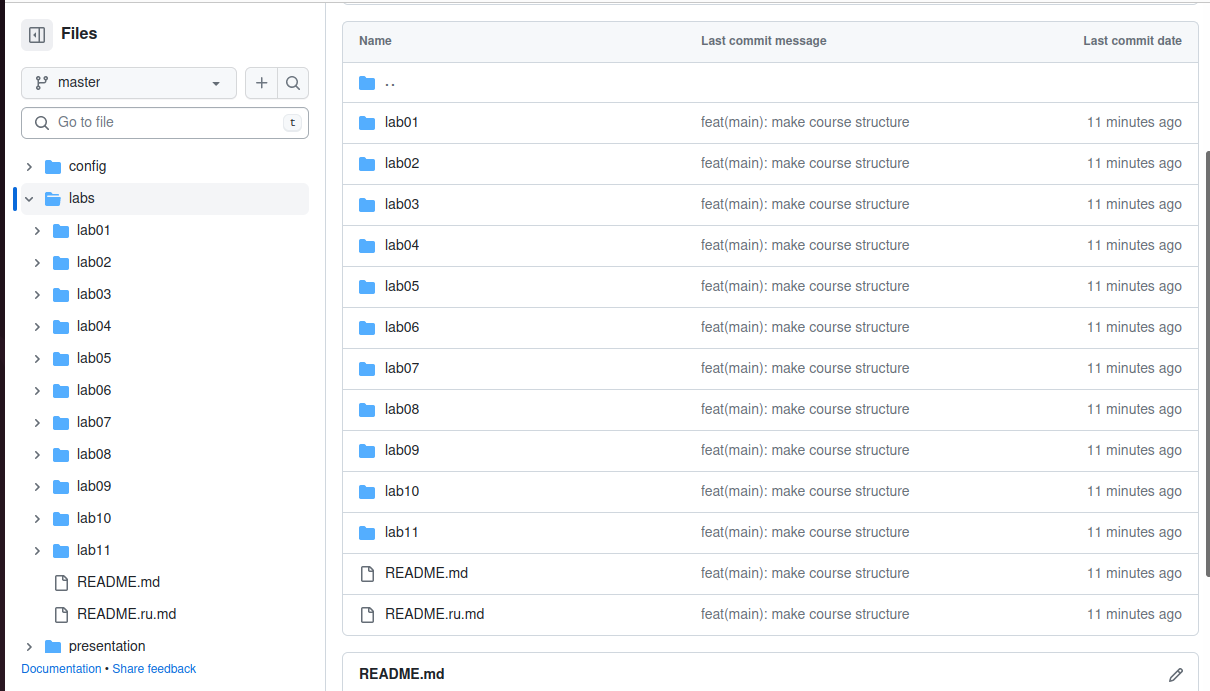


Рис. 4.12. Проверяю рабочее пространство на github

1. **Выводы**

В процессе выполнения этого лабораторного исследования я ознакомился с принципами и методами использования систем контроля изменений, а также улучшил свои практические навыки по использованию системы git