Лабораторная работа №2

Дисциплина: Архитектура компьютера

Савостин Олег

Содержание

Список иллюстраций

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. Сознание рабочего пространства и репозитория курса на основе шаблона
5. Сознание 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 распределенных системах контроля версий центральный репозиторий не является обязательным. Среди классических VCS наиболее известны CVS, Subversion, а среди распределенных — 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/(Pис. 1) и заполняю основные данные для регистрации. (Рис.2) Аккаунт готов (Рис.3).

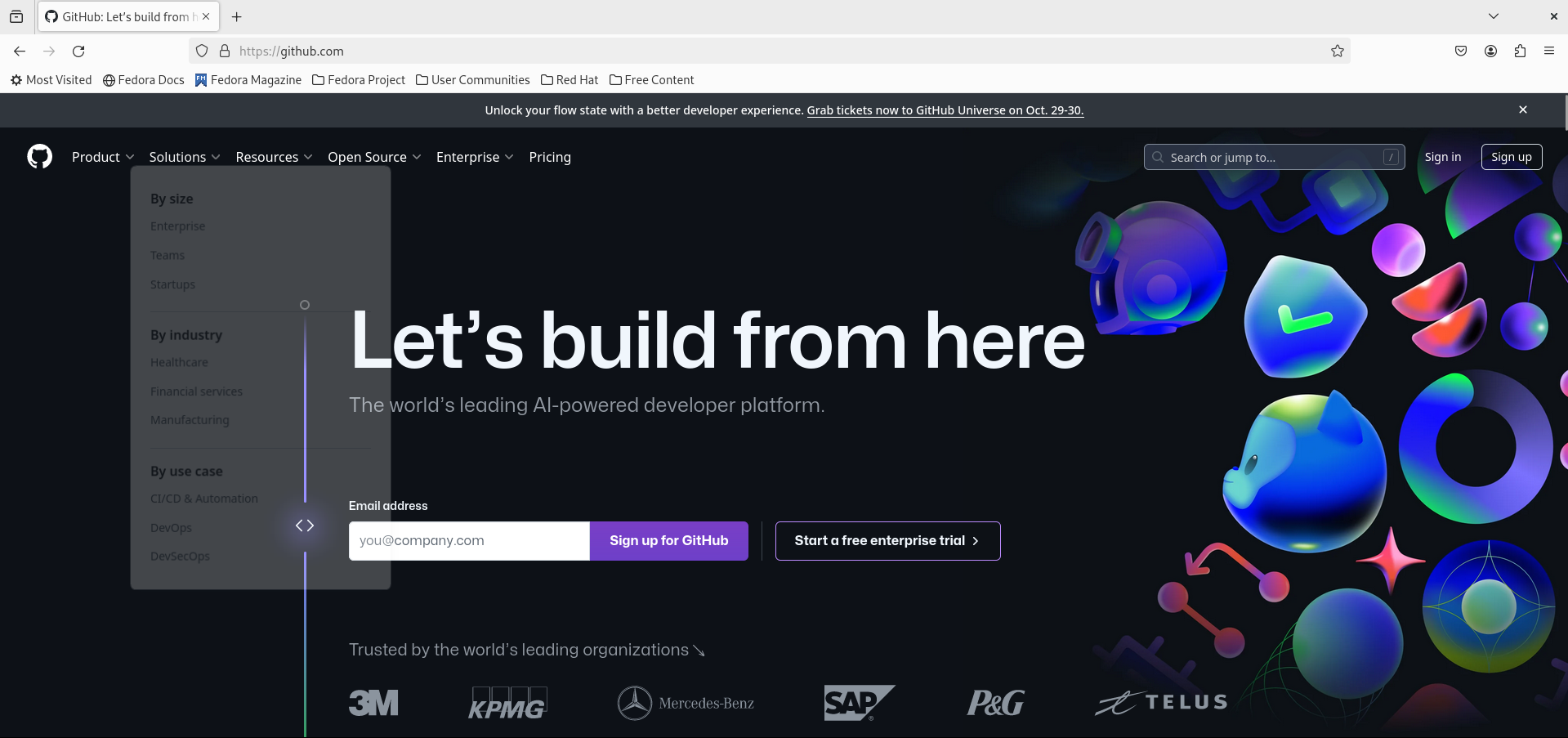


Рис. 1: Сайт github

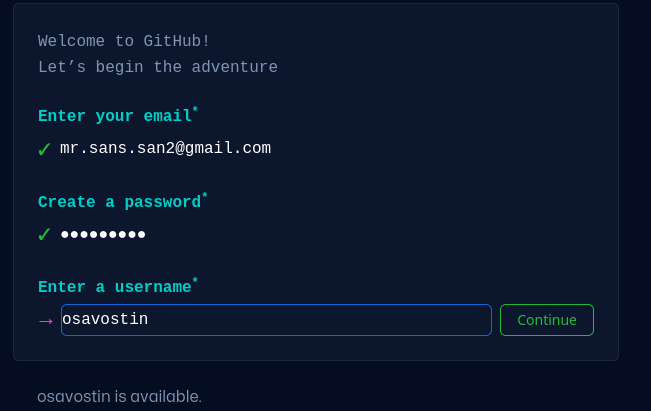


Рис. 2: Процесс регистрации на платформе.



Рис. 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).

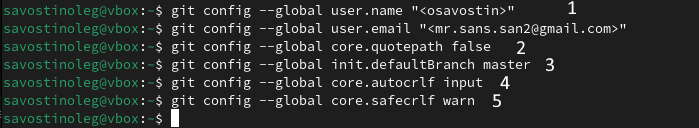


Рис. 4: Команды git

## 4.3 Создание SSH ключа

Для последующей идентификации пользователя (меня) на сервере репозиториев 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. 5)

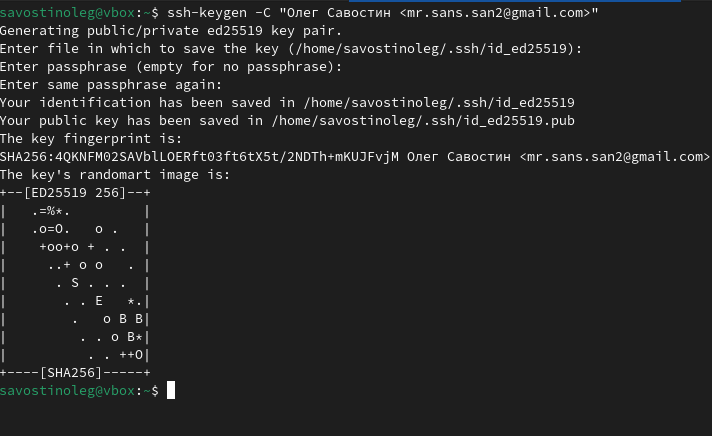


Рис. 5: Создание ключа.

Чтобы загрузить данный ключ, я захожу на github.org, захожу в свой аккаунт и захожу в настройки и выбираю опцию SSH and GPG keys (Рис. 6)

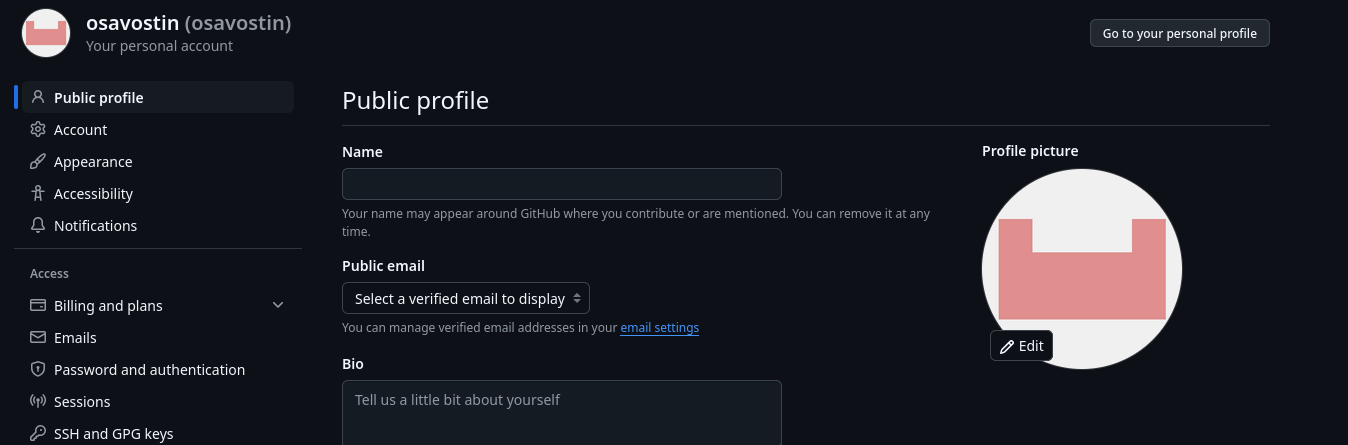


Рис. 6: Настройки аккаунта osavostin

Выбираю New SSH key. (Рис. 7)

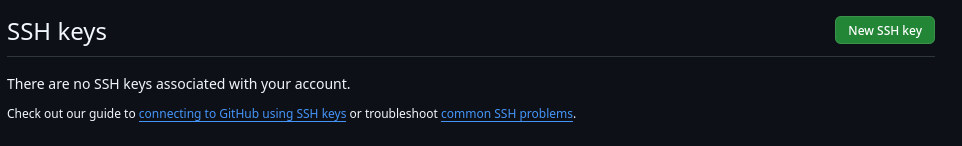


Рис. 7: New SSH key

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. 8)

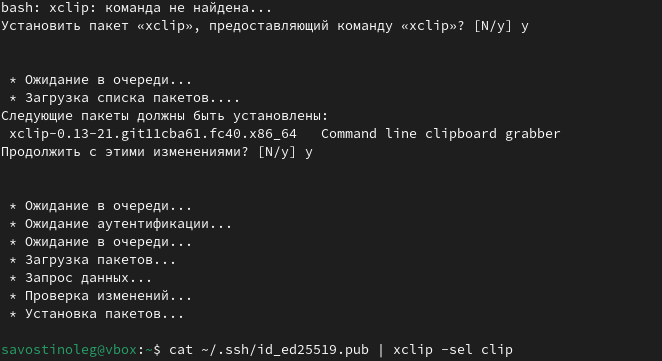


Рис. 8: Установка пакета файлов xclip и копирка данных ключа

Теперь, я вставляю скопированные данные в github (Рис. 9)

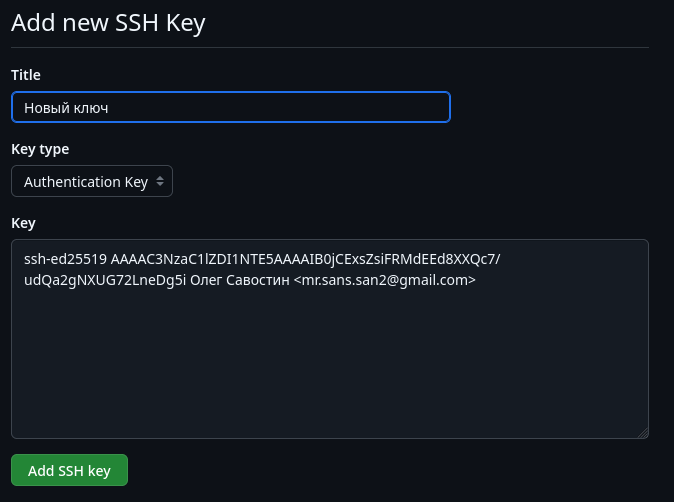


Рис. 9: Создание нового SSH ключа

## 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. 10)

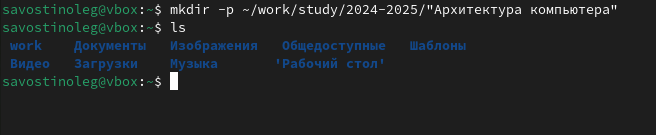


Рис. 10: Создание рабочего пространства

## 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. 11)

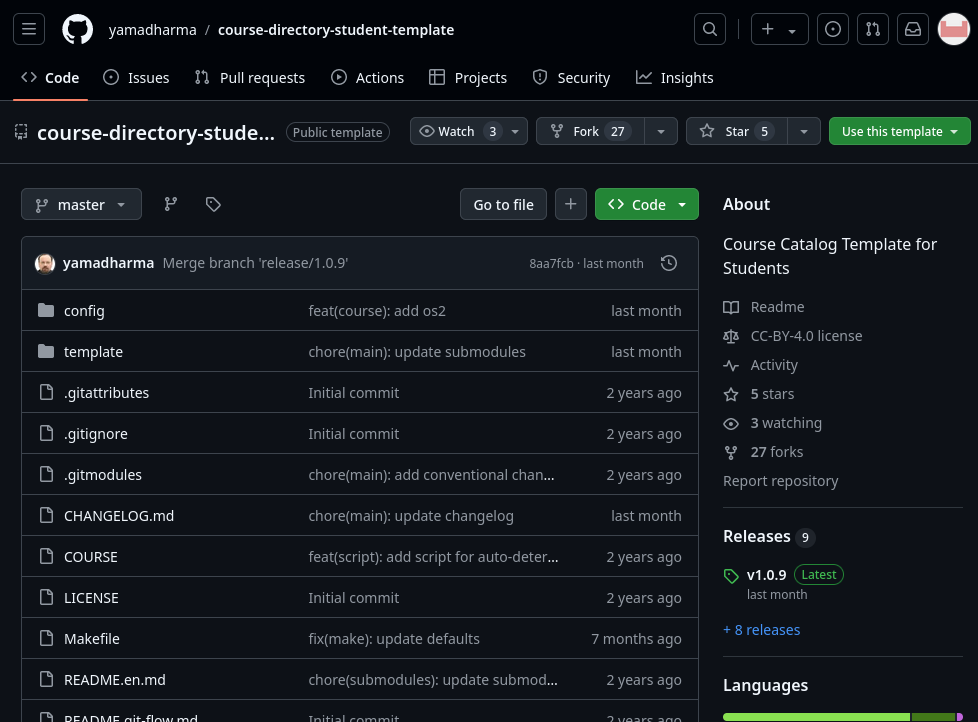


Рис. 11: Результат перехода по ссылке.

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. 12)

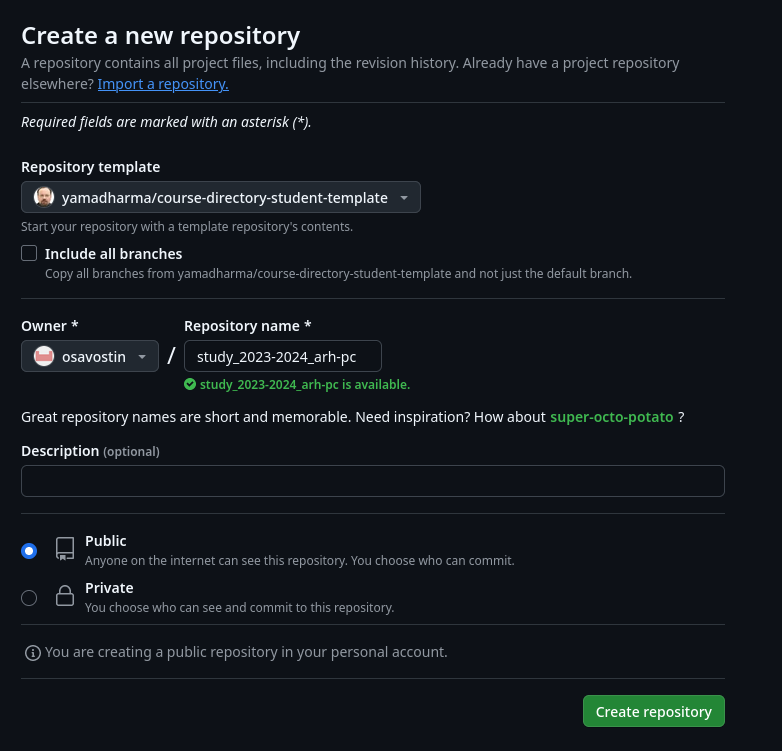


Рис. 12: Create a new repository

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. 13)

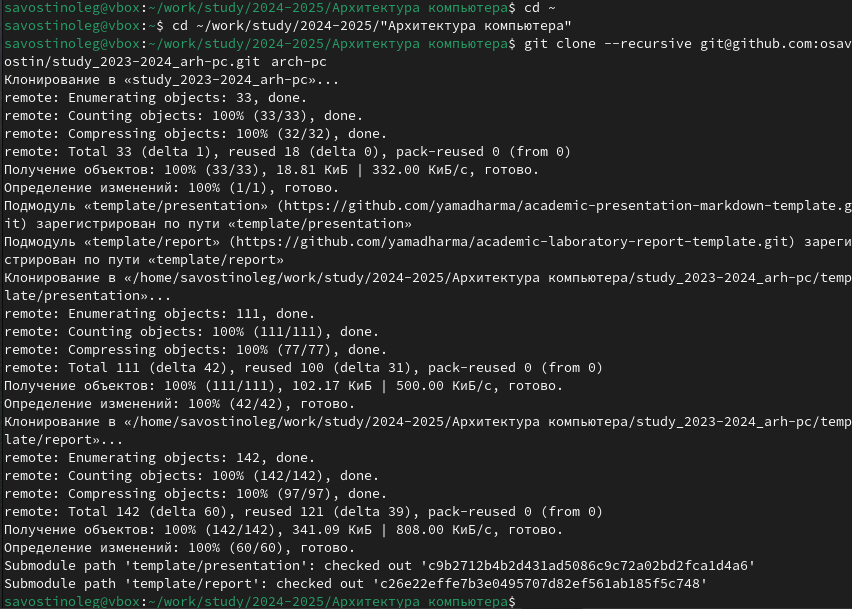


Рис. 13: Копирование репозитории в подкаталог Архитектура компьютера

Ссылка не была использована в данном случае, но это можно сделать следующим образом (Рис. 14)

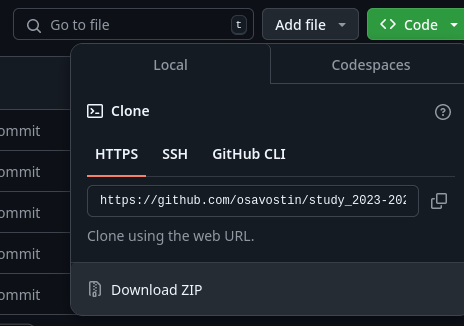


Рис. 14: Копирование ссылки.

## 4.6 Настройка каталога курса

Сперва, в терминале я перехожу в каталог курса с помощью утилиты cd в каталог arch-pc и удаляю лишний файл package.json с помощью rm, проверяю на его наличие. Затем создаю необходимые каталоги с помощью echo arch-pc > COURSE. Устанавливаю пакет make и пользуюсь этой командой (Рис. 15)

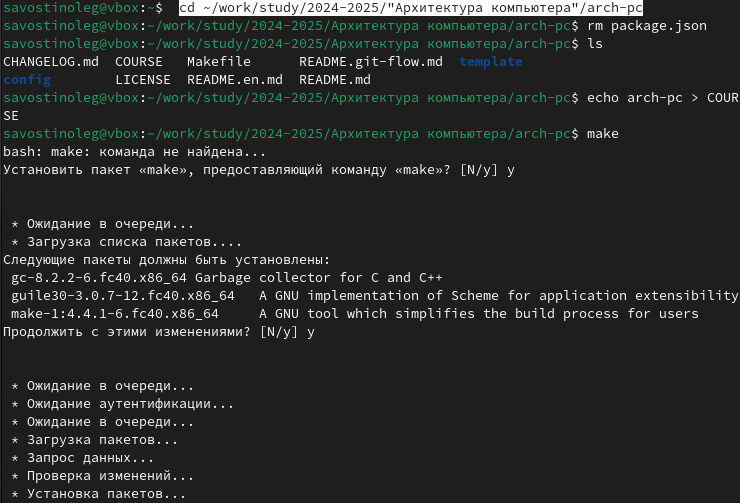


Рис. 15: rm и echo

Теперь отправляю файлы на сервер с помощью git add .; git commit -am; git push. Последняя команда отправляет все файлы на github.(Рис. 16)

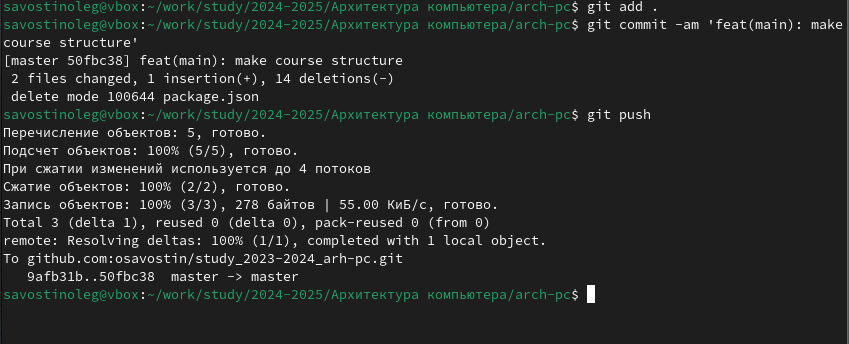


Рис. 16: Отправка файлов на github.

Проверяю если файл загрузился. (Рис. 17)

Рис. 17: COURSE

Рис. 17: COURSE

## 4.7 Задания для самостоятельной работы

1. Создаю отчет по выполнению лабораторной работы в labs/lab02/report/ с помощью утилиты touch(Рис. 18)

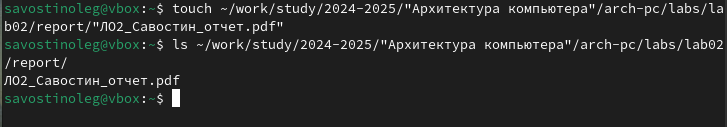


Рис. 18: Создание отчета и проверка на наличие

1. 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. 19)

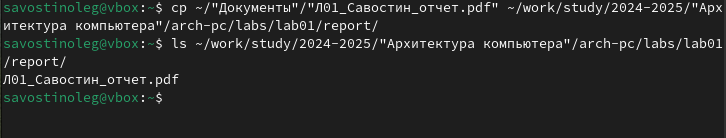


Рис. 19: Копирка файла в lab01/report

1. Теперь загружаю файлы на github используя git add .; git commit -m; git push.(Рис. 20)

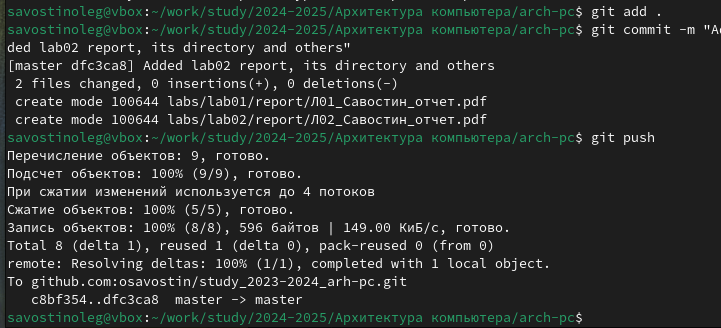


Рис. 20: Загрузка на github

Как итог, всё прошло успешно. (Рис. 21)

Рис. 21: labs в github

Рис. 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](https://esystem.rudn.ru/pluginfile.php/2089527/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%962.%20%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8F%20%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B9%20Git.pdf) РУДН