**Отчёт по лабораторной работе №2**

**Дисциплина: архитектура компьютера**

Беспутин Глеб Антонович

**Содержание**

[1 Цель работы 4](#_Toc13110)

[2 Задание 5](#_Toc13111)

[3 Теоретическое введение 6](#_Toc13112)

[4 Выполнение лабораторной работы 8](#_Toc13113)

[4.1 Настройка GitHub 8](#_Toc13114)

[4.2 Базовая настройка Git 9](#_Toc13115)

[4.3 Создание SSH-ключа 10](#_Toc13116)

[4.4 Создание рабочего пространства и репозитория курса на основе шаблона 13](#_Toc13117)

[4.5 Создание репозитория курса на основе шаблона 14](#_Toc13118)

[4.6 Настройка каталога курса 16](#_Toc13119)

[4.7 Выполнение заданий для самостоятельной работы 19](#_Toc13120)

[5 Выводы 22](#_Toc13121)

[6 Список литературы 23](#_Toc13122)

3

# Цель работы

Целью данной работы является изучить идеологию и применение средств контроля версий, а также приобрести практические навыки по работе с системой git.

# 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.

# Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при ра- боте нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен до- ступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, про- изведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется. В классических системах контроля версий используется централизованная модель, предполагающая наличие еди- ного репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник про- екта (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 уда- ляются 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 по- следовательными версиями, что позволяет уменьшить объём хранимых данных. Системы контроля версий поддерживают возможность отслеживания и разреше- ния 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 заблокиро- вать 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 измене- ний 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 распреде- лённых — 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.

# Выполнение лабораторной работы

## Настройка GitHub

Создаю учетную запись на сайте GitHub (рис. 4.1). Далее я заполнил основные данные учетной записи.

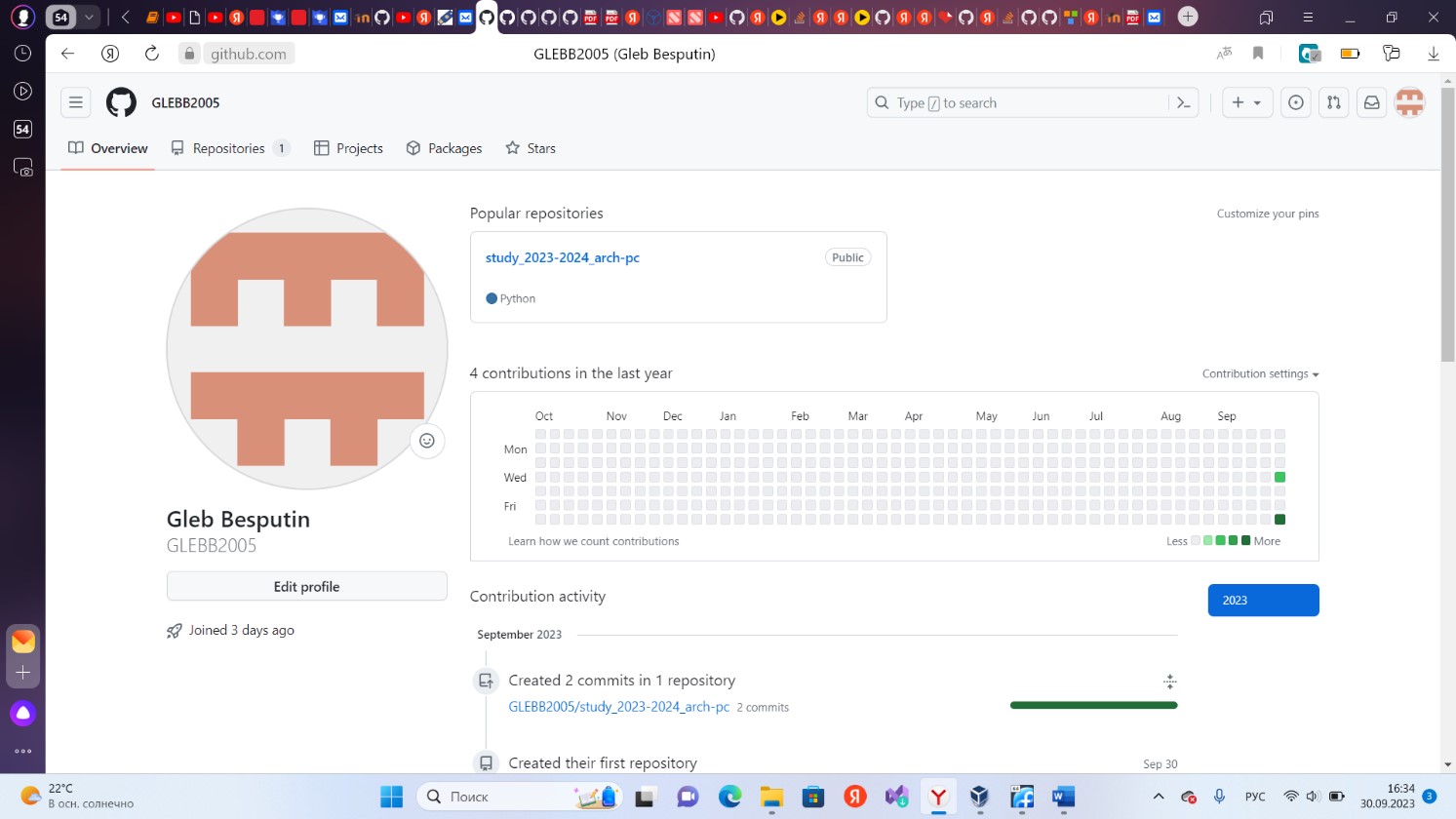


Рис. 4.1: Заполнение данных учетной записи GitHub

Аккаунт создан (рис. 4.2).

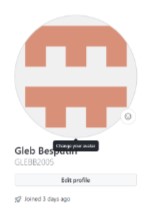


Рис. 4.2: Аккаунт GitHub

## Базовая настройка Git

Открываю виртуальную машину, затем открываю терминал и делаю пред- варительную конфигурацию git. Ввожу команду git config –global user.name “”, указывая свое имя и команду git config –global user.email “work@mail”, указывая в ней электронную почту владельца, то есть мою (рис. 4.3).

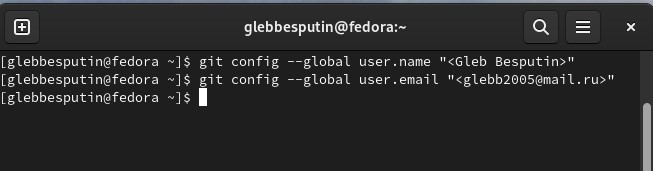


Рис. 4.3: Предварительная конфигурация git

Настраиваю utf-8 в выводе сообщений git для корректного отображения сим- волов (рис. 4.4).



Рис. 4.4: Настройка кодировки

Задаю имя «master» для начальной ветки (рис. 4.5).



Рис. 4.5: Создание имени для начальной ветки

Задаю параметр autocrlf со значением input, так как я работаю в системе Linux, чтобы конвертировать CRLF в LF только при коммитах (рис. 4.6). CR и LF – это сим- волы, которые можно использовать для обозначения разрыва строки в текстовых файлах.



Рис. 4.6: Параметр autocrlf

Задаю параметр safecrlf со значением warn, так Git будет проверять преоб- разование на обратимость (рис. 4.7). При значении warn Git только выведет предупреждение, но будет принимать необратимые конвертации.



Рис. 4.7: Параметр safecrlf

## Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозиториев необ- ходимо 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. 4.8). Ключ автоматически сохранится в

каталоге ~/.ssh/.

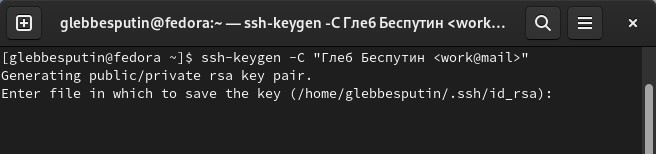


Рис. 4.8: Генерация SSH-ключа

Xclip – утилита, позволяющая скопировать любой текст через терминал. Ока- зывается, в дистрибутиве Linux Kali ее сначала надо установить. Устанавливаю xclip с помощью команды apt-get install с ключом -y отимени суперпользователя, введя в начале команды sudo (рис. 4.9).

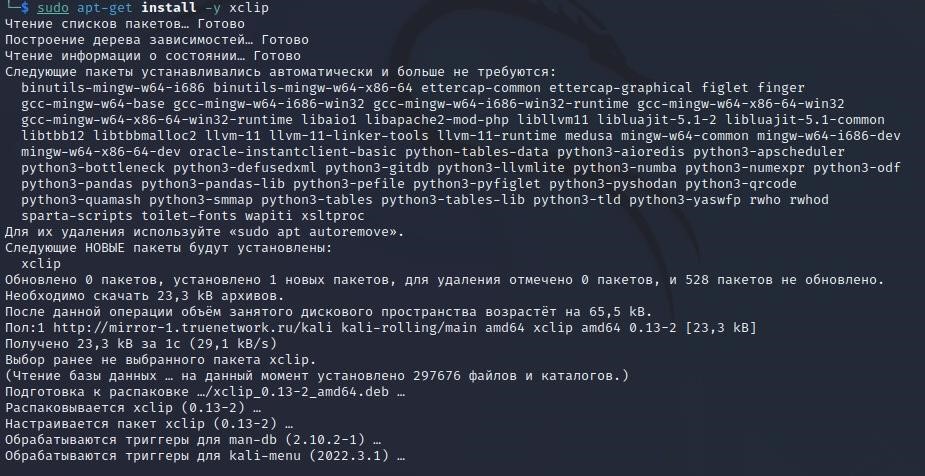


Рис. 4.9: Установка утилиты xclip

Копирую открытый ключ из директории, в которой он был сохранен, с помо- щью утилиты xclip (рис. 4.10).

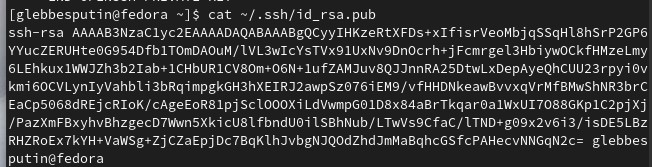


Рис. 4.10: Копирование содержимого файла

Открываю браузер, захожу на сайт GitHub. Открываю свой профиль и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key» (рис. 4.11).

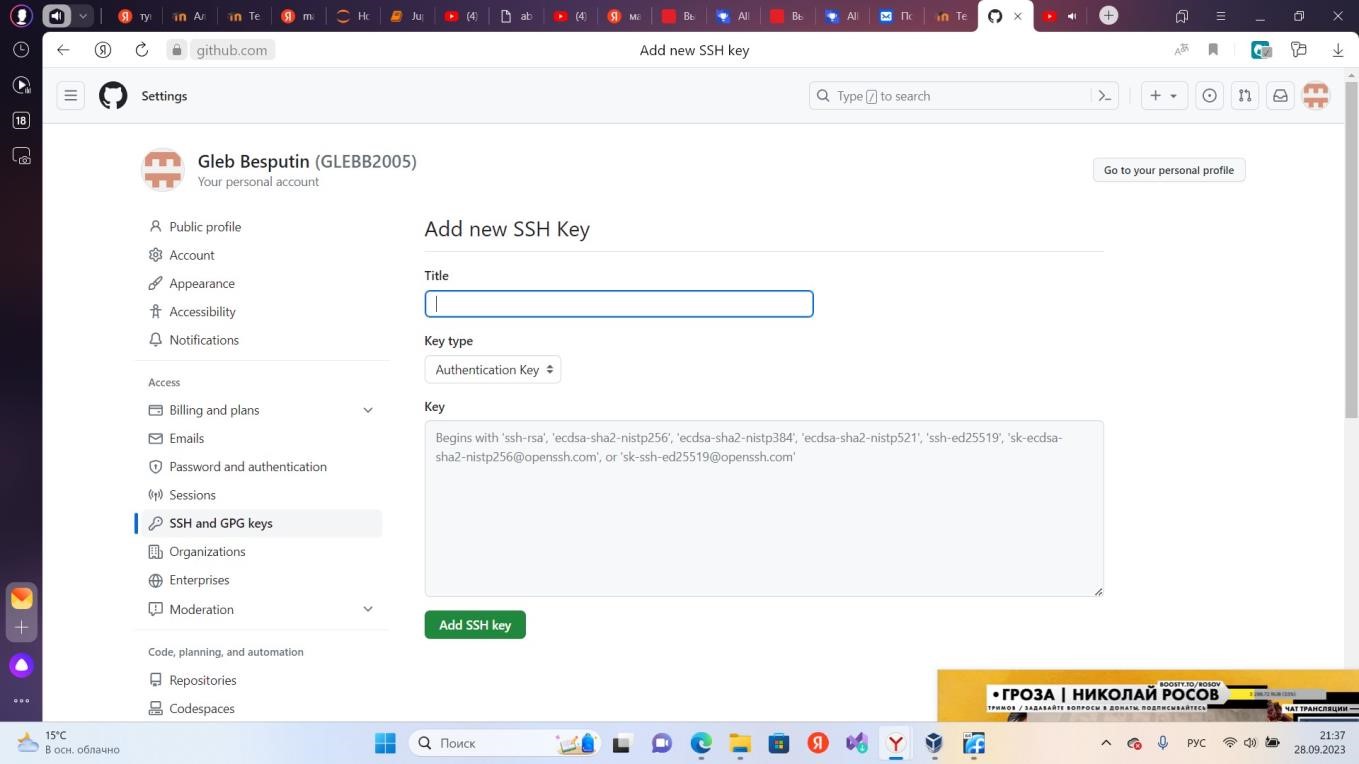


Рис. 4.11: Окно SSH and GPG keys

Вставляю скопированный ключ в поле «Key». В поле Title указываю имя для ключа. Нажимаю «Add SSH-key», чтобы завершить добавление ключа (рис. 4.12).

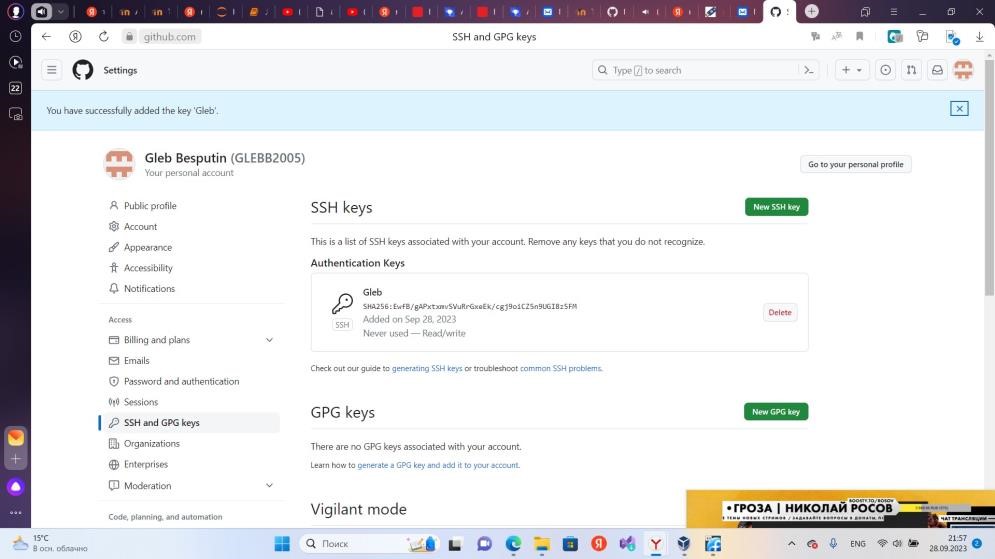


Рис. 4.12: Добавление ключа

## Создание рабочего пространства и репозитория курса на основе шаблона

Закрываю браузер, открываю терминал. Создаю директорию, рабочее про- странство, с помощью утилиты mkdir, блягодаря ключу -p создаю все директории после домашней ~/work/study/2022-2023/“Архитектура компьютера” рекурсивно.

(рис. 4.13).



Рис. 4.13: Создание рабочего пространства

## Создание репозитория курса на основе шаблона

В браузере перехожу на страницу репозитория с шаблоном курса по адресу https://github.com/yamadharma/course-directory-student-template. Далее выбираю

«Use this template», чтобы использовать этот шаблон для своего репозитория (рис.

4.14).

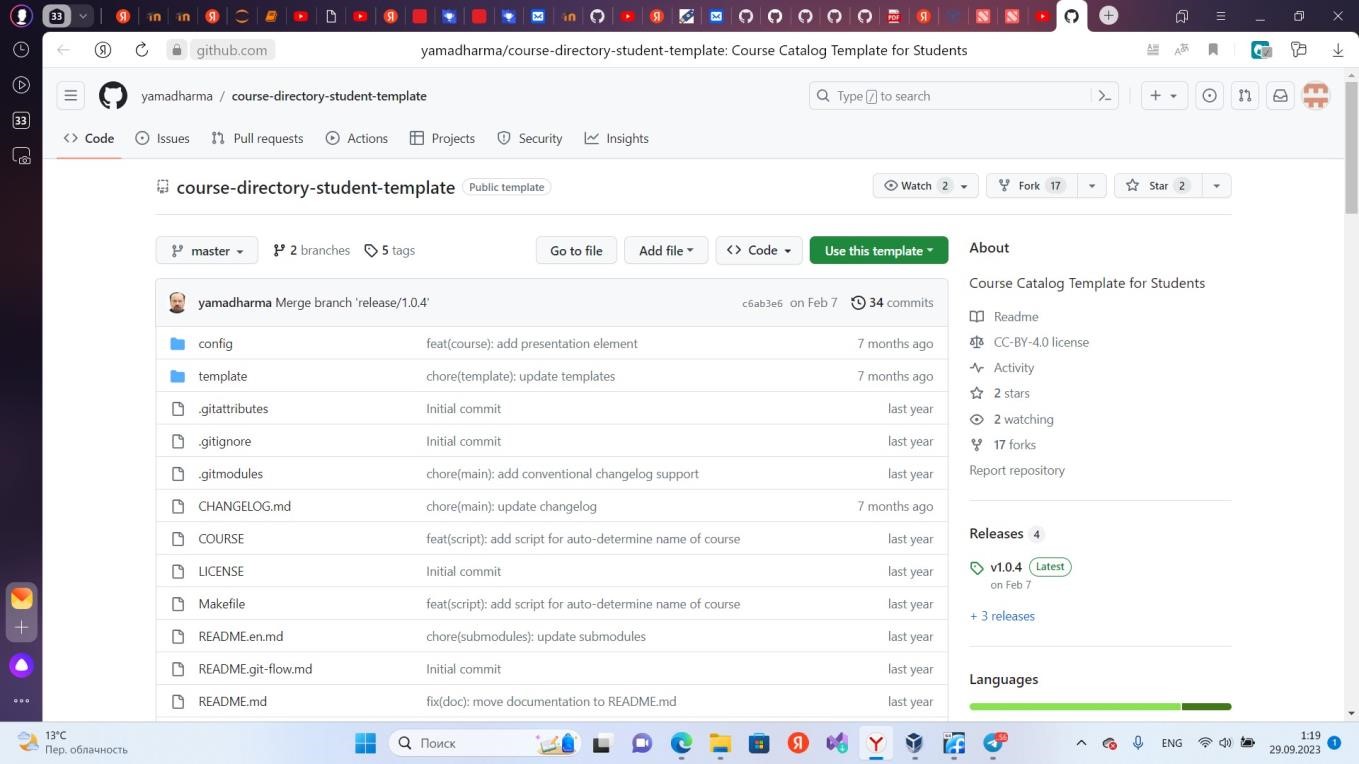


Рис. 4.14: Страница шаблона для репозитория

В открывшемся окне задаю имя репозитория (Repository name): study\_2022–2023\_arh- pc и создаю репозиторий, нажимаю на кнопку «Create repository from template» (рис. 4.15).

Рис. 4.15: Окно создания репозитория

Репозиторий создан (рис. 4.16).

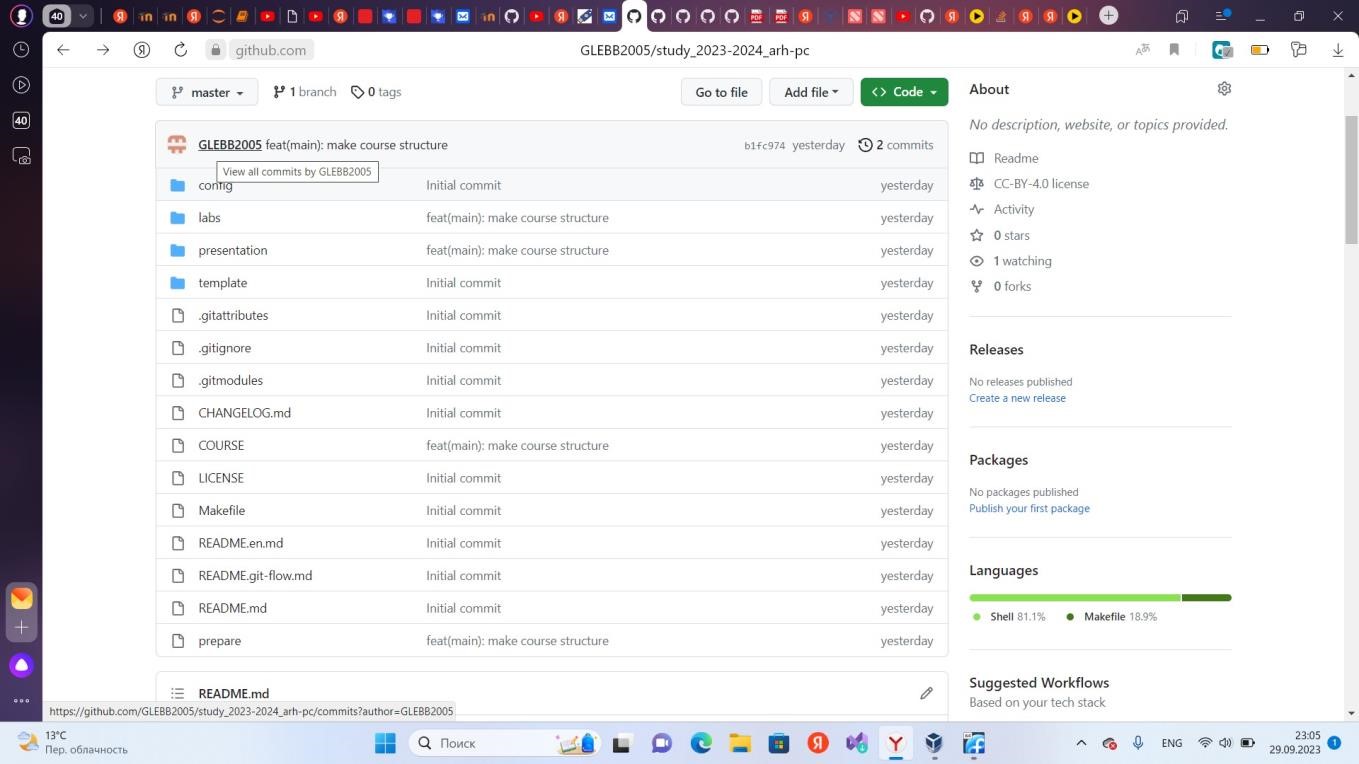


Рис. 4.16: Созданный репозиторий

Через терминал перехожу в созданный каталог курса с помощью утилиты cd

(рис. 4.17).



Рис. 4.17: Перемещение между директориями

Клонирую созданный репозиторий с помощью команды git clone –recursive git@github.com:/study\_2022–2023\_arh-pc.git arch-pc (рис. 4.18).

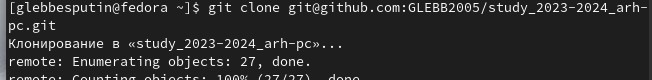


Рис. 4.18: Клонирование репозитория

Копирую ссылку для клонирования на странице созданного репозитория, сна- чала перейдя в окно «code», далее выбрав в окне вкладку «SSH» (рис. 4.19).

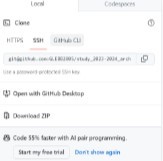


Рис. 4.19: Окно с ссылкой для копирования репозитория

## Настройка каталога курса

Перехожу в каталог arch-pc с помощью утилиты cd (рис. 4.20).



Рис. 4.20: Перемещение между директориями

Удаляю лишние файлы с помощью утилиты rm (рис. 4.21).



Рис. 4.21: Удаление файлов

Создаю необходимые каталоги (рис. 4.22).



Рис. 4.22: Создание каталогов

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. 4.23).

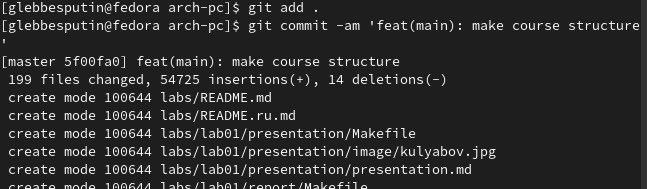


Рис. 4.23: Добавление и сохранение изменений на сервере

Отправляю все на сервер с помощью push (рис. 4.24).

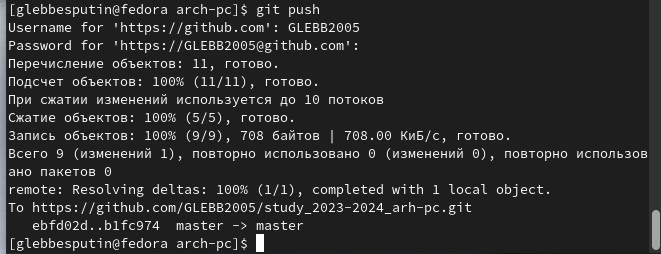


Рис. 4.24: Выгрузка изменений на сервер

Проверяю правильность выполнения работы сначала на самом сайте GitHub

(рис. 4.25).

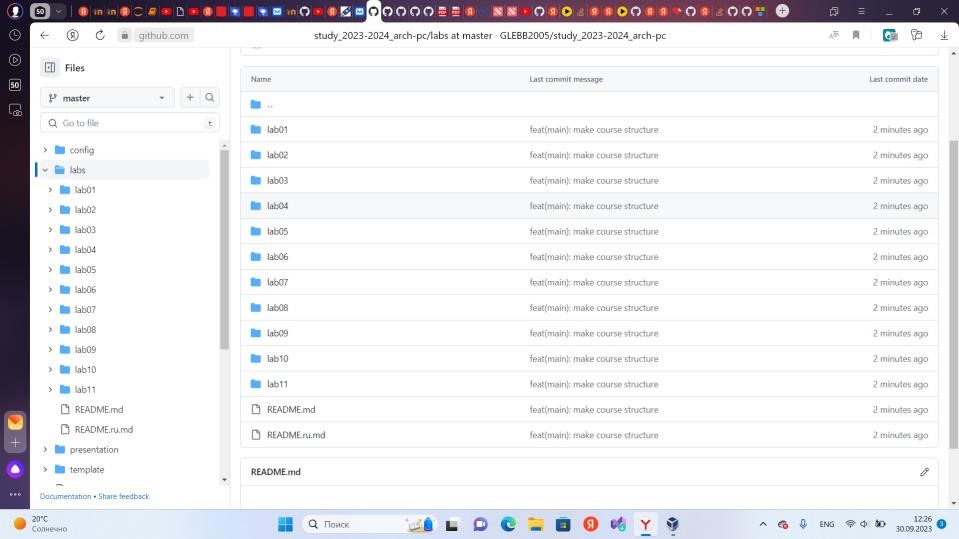


Рис. 4.25: Страница репозитория

## Выполнение заданий для самостоятельной работы

1. Создаю отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs>lab02>report).

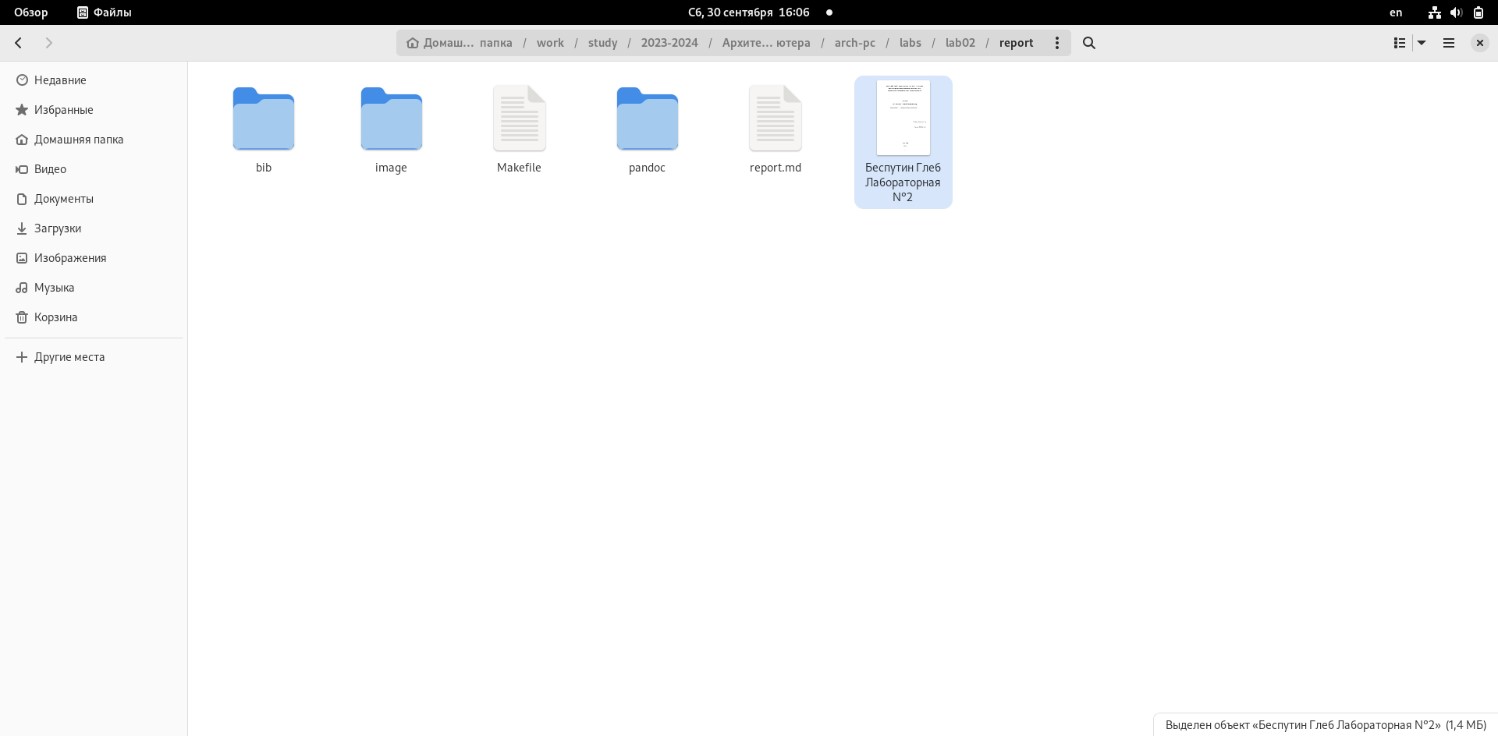


Рис. 4.26: Создание файла

1. Копирую отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства.

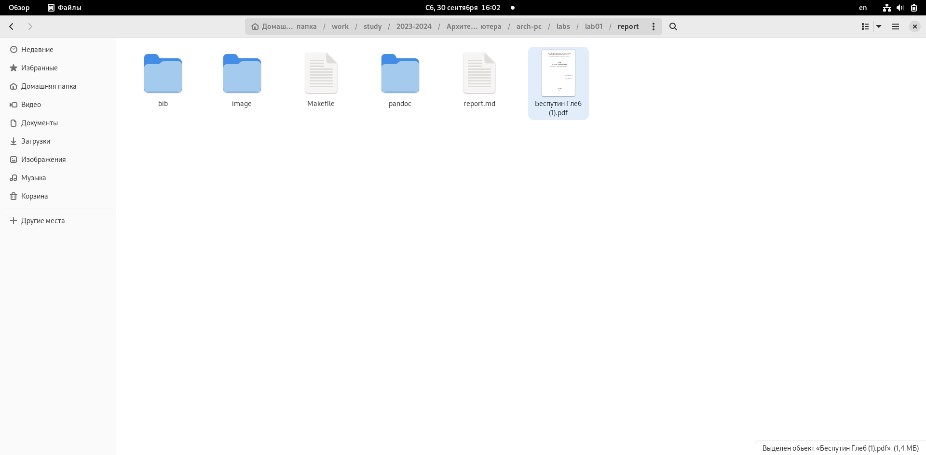


Рис. 4.27: Копирую

1. Загружаю файлы на гитхаб. (ссылка на гитхаб будет прикреплена, там можно будет увидеть эти файлы)

# Выводы

При выполнении данной лабораторной работы я изучил идеологию и при- менение средств контроля версий, а также приобрел практические навыки по

работе с системой git.

# Список литературы

1. [Архитектура ЭВМ](https://esystem.rudn.ru/pluginfile.php/1584622/mod_resource/content/1/%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%963.pdf)
2. [Git - gitattributes Документация](https://git-scm.com/docs/gitattributes)