**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: Краснова К. Г.

Группа: НКАбд-05-24

**МОСКВА**

2024 г.

Содержание

[1 Цель работы 4](#_Toc178427956)

[2 Задание 5](#_Toc178427957)

[3 Теоретическое введение 6](#_Toc178427958)

[4 Выполнение лабораторной работы 8](#_Toc178427959)

[4.1 Настройка GitHub 8](#_Toc178427960)

[4.2 Базовая настройка Git 8](#_Toc178427961)

[4.3 Создание SSH-ключа 8](#_Toc178427962)

[4.4 Создание рабочего пространства и репозитория курса на основе шаблона 10](#_Toc178427963)

[4.5 Создание репозитория курса на основе шаблона 10](#_Toc178427964)

[4.6 Настройка каталога курса 13](#_Toc178427965)

[4.7 Выполнение заданий для самостоятельной работы 15](#_Toc178427966)

[5 Выводы 17](#_Toc178427967)

[6 Список литературы 18](#_Toc178427968)

**Список иллюстраций**

[Рис. 1. Аккаунт GitHub 8](#_Toc178427286)

[Рис. 2. Предварительная конфигурация git 8](#_Toc178427287)

[Рис. 3. Базовая настройка git 8](#_Toc178427288)

[Рис. 4. Генерация SSH-ключа 9](#_Toc178427289)

[Рис. 5. Копирование ключа 9](#_Toc178427290)

[Рис. 6. Окно SSH and GPG keys 9](#_Toc178427291)

[Рис. 7. Добавление ключа 10](#_Toc178427292)

[Рис. 8. Создание рабочего пространства 10](#_Toc178427293)

[Рис. 9. Страница шаблона для репозитория 11](#_Toc178427294)

[Рис. 10. Окно создания репозитория 12](#_Toc178427295)

[Рис. 11. Созданный репозиторий 12](#_Toc178427296)

[Рис. 12. Клонирование репозитория 13](#_Toc178427297)

[Рис. 13. Ссылка для клонирования репозитория 13](#_Toc178427298)

[Рис. 14. Перемещение между директориями 13](#_Toc178427299)

[Рис. 15. Удаление файла 13](#_Toc178427300)

[Рис. 16. Добавление и сохранение изменений на сервере 14](#_Toc178427301)

[Рис. 17. Отправка изменений на сервер 14](#_Toc178427302)

[Рис. 18. Проверка выполнения 14](#_Toc178427303)

[Рис. 19. Проверка выполнения 15](#_Toc178427304)

[Рис. 20. Создание файла 15](#_Toc178427305)

[Рис. 21. Перемещение между директориями 15](#_Toc178427306)

[Рис. 22. Проверка местонахождения файла 15](#_Toc178427307)

[Рис. 23. Копирование файла 15](#_Toc178427308)

[Рис. 24. Загрузка файлов на GitHub 16](#_Toc178427309)

[Рис. 25. Расположение каталога 16](#_Toc178427310)

# 1 Цель работы

Целью данной работы является изучить идеологию и применение средств контроля версий, а также 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 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 (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.

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. Аккаунт создан (рис. 1).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 1. Аккаунт GitHub

## 4.2 Базовая настройка Git

Делаю предварительную конфигурацию git. Ввожу команду git config –global user.name <name> и git config –global user.email <work@mail>, указывая свое имя и свою почту (рис. 2).



Рис. 2. Предварительная конфигурация git

Настраиваю utf-8 в выводе сообщений git, задаю имя начальной ветки «master», задаю параметры autorclf и saferclf (рис. 3).

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рис. 3. Базовая настройка git

## 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 свое имя и почту (рис.4).

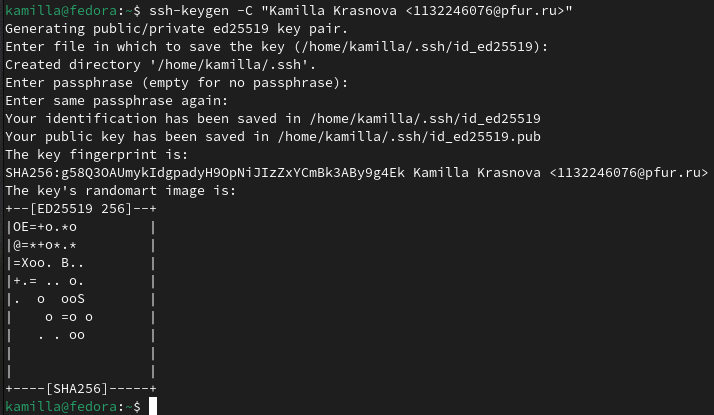


Рис. 4. Генерация SSH-ключа

Копирую открытый ключ (рис. 5).



Рис. 5. Копирование ключа

Открываю свой профиль и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key» (рис. 6).Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рис. 6. Окно SSH and GPG keys

Вставляю скопированный ключ в поле «Key». В поле Title указываю имя для ключа. Нажимаю «Add SSH-key», чтобы добавить ключ (рис. 7).

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рис. 7. Добавление ключа

## 4.4 Создание рабочего пространства и репозитория курса на основе шаблона

Создаю каталог для предмета «Архитектура компьютера» с помощью mkdir и ключа -p. Проверяю с помощью ls правильность выполнения команды (рис.8).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 8. Создание рабочего пространства

## 4.5 Создание репозитория курса на основе шаблона

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. 9).

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Рис. 9. Страница шаблона для репозитория

В открывшемся окне задаю имя репозитория: study\_2024–2025\_arh-pc и создаю репозиторий, нажимаю на кнопку «Create repository from template» (рис. 10).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 10. Окно создания репозитория

Созданный репозиторий (рис. 11).

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рис. 11. Созданный репозиторий

Перехожу в каталог курса и клонирую созданный репозиторий (рис. 12).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 12. Клонирование репозитория

Копирую ссылку для клонирования на странице созданного репозитория, сначала перейдя в окно «Сode», далее выбрав в окне вкладку «SSH» (рис. 13).

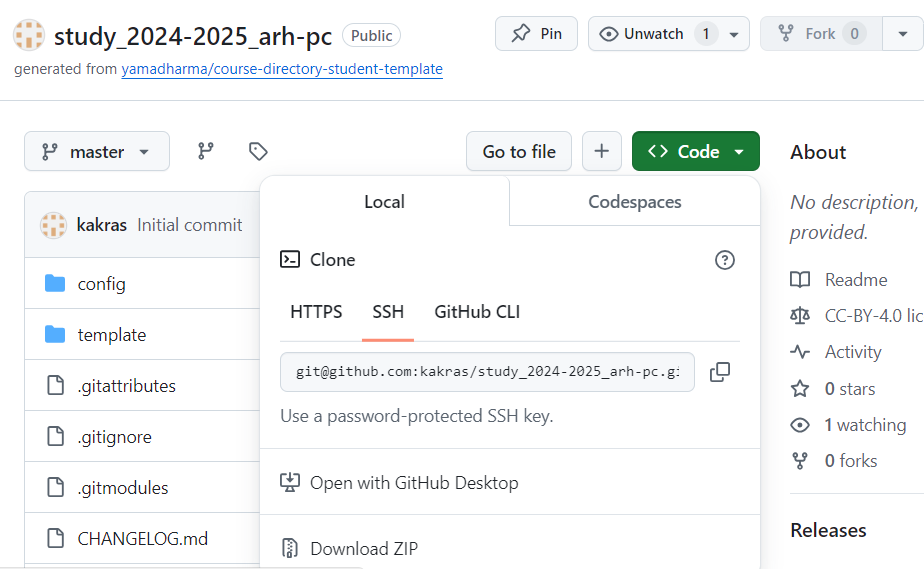


Рис. 13. Ссылка для клонирования репозитория

## 4.6 Настройка каталога курса

Перехожу в каталог arch-pc с помощью cd (рис. 14).



Рис. 14. Перемещение между директориями

Удаляю лишние файлы (рис. 15).



Рис. 15. Удаление файла

Создаю необходимые каталоги. Затем добавляю созданные каталоги с помощью git add и сохраняю изменения на сервере(рис. 16).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 16. Добавление и сохранение изменений на сервере

Отправляю все на сервер (рис. 17).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 17. Отправка изменений на сервер

Проверяю правильность выполнения работы сначала на странице GitHub (рис. 18), а затем в локальном репозитории (рис. 19).

Изображение выглядит как текст, линия, Шрифт, число

Автоматически созданное описание

Рис. 18. Проверка выполнения



Рис. 19. Проверка выполнения

## 4.7 Выполнение заданий для самостоятельной работы

1. Создаю в директории labs/lab02/report файл для отчета по второй лабораторной работе с помощью touch. Проверяю правильность выполнения команды (рис. 20).

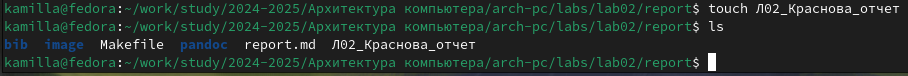


Рис. 20. Создание файла

2. Перехожу из подкаталога lab02/report в подкаталог lab01/report с помощью cd (рис.21).



Рис. 21. Перемещение между директориями

Проверяю местонахождение файла с отчетом по первой лабораторной работе (рис.22.)



Рис. 22. Проверка местонахождения файла

Копирую первую лабораторную работу с помощью cp и проверяю правильность выполнения команды с помощью ls (рис. 23).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 23. Копирование файла

3. Добавляю с помощью git add файл Л01\_Краснова\_отчет, сохраняю изменения на сервере и отправляю сохраненные изменения в репозиторий (рис. 24).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 24. Загрузка файлов на GitHub

Проверяю на GitHub правильность выполнения заданий. Файл с отчетом находится в соответствующем каталоге репозитория (рис. 25).

Изображение выглядит как текст, Шрифт, число, снимок экрана

Автоматически созданное описание

Рис. 25. Расположение каталога

То же самое проделываю для файла Л02\_Краснова\_отчет.pdf.

# 5 Выводы

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

# 6 Список литературы

1. [Git – gitattributes Документация](https://git-scm.com/docs/gitattributes)