Отчет по лабораторной работе №2

Операционные системы

Мустафина Аделя Юрисовна

Содержание

Список иллюстраций

Список таблиц

# 1 Цель работы

Изучить идеологию и применение средств контроля версий и освоить умения по работе с git.

# 2 Задание

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Настроить подписи git.
5. Зарегистрироваться на Github.
6. Создать локальный каталог для выполнения заданий по предмету.

# 3 Выполнение лабораторной работы

## 3.1 Установка программного обеспечения

Устанавливаю программное обеспечение через терминал с помощью команд dnf install git и dnf install gh (рис. 1).

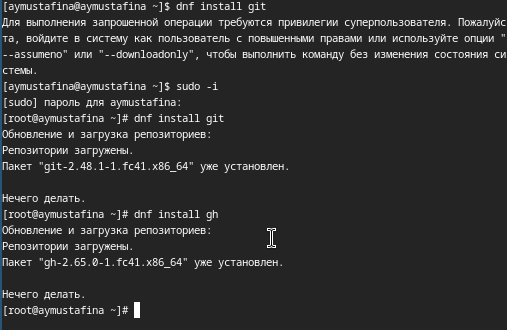


Рис. 1: Установка

## 3.2 Базовая настройка git

Задаю имя и email владельца репозитория с помощью команд: git config –global user.name “Name Surname” git config –global user.email “work@mail” (рис. 2).

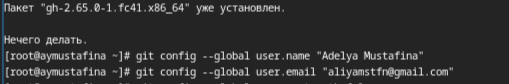


Рис. 2: Имя и email

Настроим utf-8 в выводе сообщений git: git config –global core.quotepath false

Зададим имя начальной ветки (будем называть её master): git config –global init.defaultBranch master

Параметр autocrlf: git config –global core.autocrlf input

Параметр safecrlf: git config –global core.safecrlf warn (рис. 3).

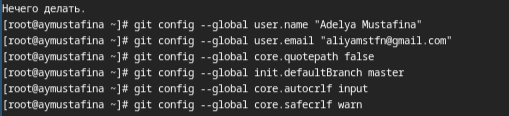


Рис. 3: Базовая настройка

## 3.3 Создание ключа ssh

по алгоритму rsa с ключём размером 4096 бит: ssh-keygen -t rsa -b 4096 (рис. 4).

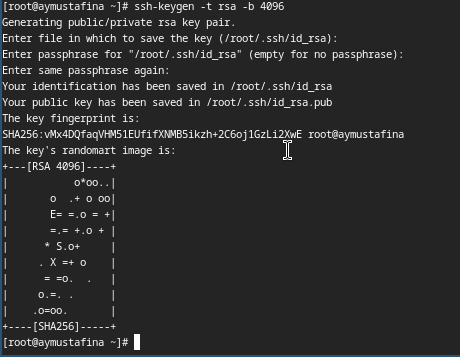


Рис. 4: Создание ключа ssh по rsa

по алгоритму ed25519: ssh-keygen -t ed25519 (рис. 5).

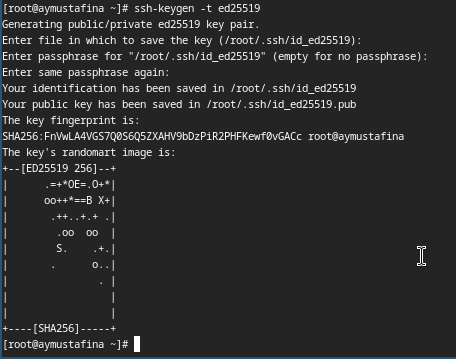


Рис. 5: Создание ключа ssh по ed255519

## 3.4 Создание ключа pgp

Генерирую ключ gpg –full-generate-key

Из предложенных опций выбираю: тип RSA and RSA; размер 4096; выбрала срок действия; значение по умолчанию — 0 (срок действия не истекает никогда). GPG запросит личную информацию, которая сохранится в ключе: Имя. Адрес электронной почты. При вводе email убеждаюсь, что он соответствует адресу, используемому на GitHub. Комментарий. Нажимаю клавишу ввода, чтобы оставить это поле пустым. (рис. 6).



Рис. 6: Генерирование ключа pgp

Ввожу фразу-пароль для защиты ключа (рис. 7).

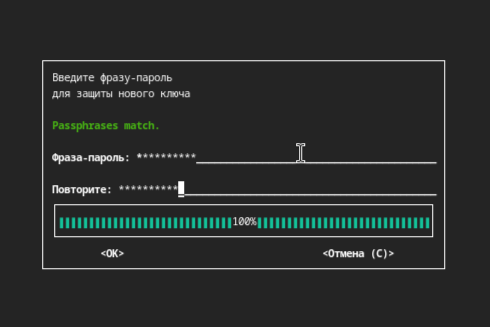


Рис. 7: Фраза-пароль

Открытый и секретный ключ созданы (рис. 8).

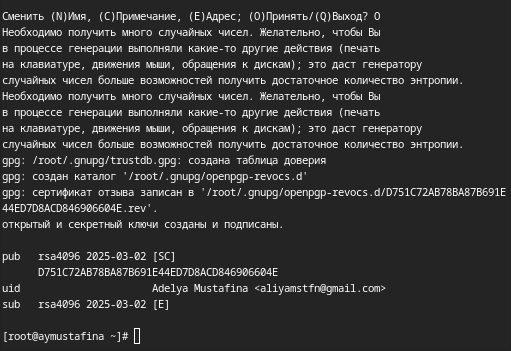


Рис. 8: Создание ключей

## 3.5 Настройка github

У меня уже был создан аккаунт на github, основные данные аккаунта заполнены (рис. 9).

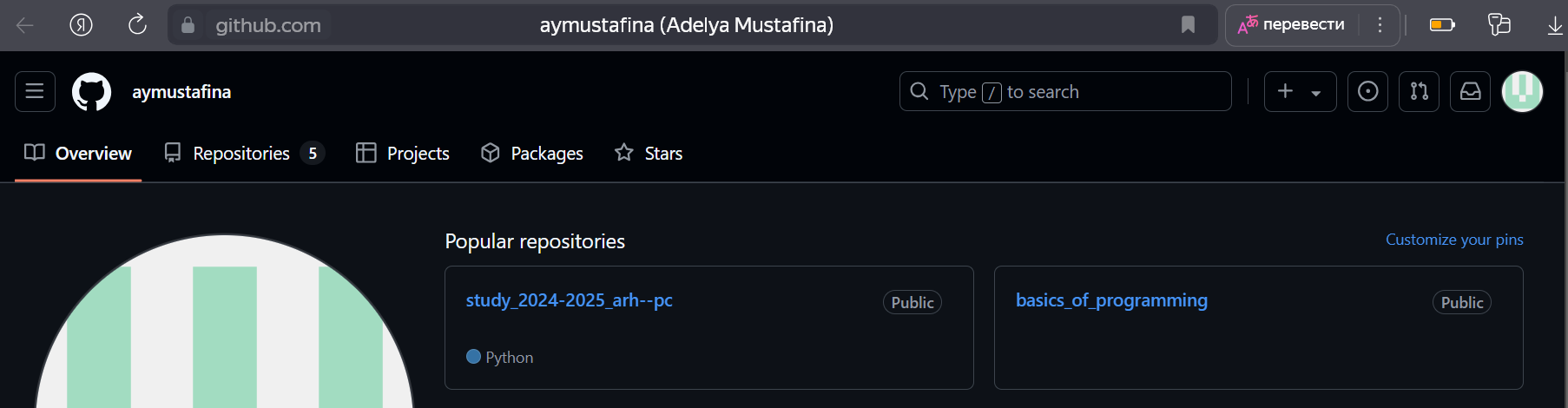


Рис. 9: Настройка github

## 3.6 Добавление PGP ключа в GitHub

Вывожу список ключей и копирую отпечаток приватного ключа: gpg –list-secret-keys –keyid-format LONG

Отпечаток ключа — это последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа.(рис. 10).

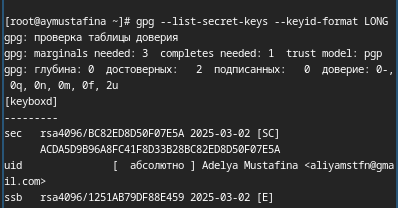


Рис. 10: Вывод ключей

Формат строки:

sec Алгоритм/Отпечаток\_ключа Дата\_создания [Флаги] [Годен\_до] ID\_ключа Копирую сгенерированный PGP ключ в буфер обмена: gpg –armor –export | xclip -sel cli (рис. 11).

Рис. 11: Копирование ключей

Рис. 11: Копирование ключей

Перехожу в настройки Github, нажимаю кнопку New GPG key и вставляю полученный ключ в поле ввода (рис. 12).

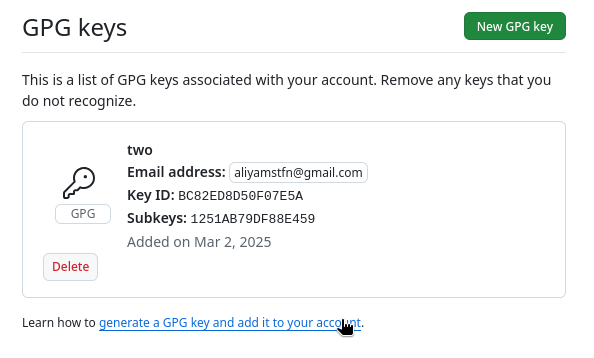


Рис. 12: New GPG key

## 3.7 Настройка автоматических подписей коммитов git

Используя введёный email, укажите Git применять его при подписи коммитов:

git config –global user.signingkey git config –global commit.gpgsign true git config –global gpg.program $(which gpg2) (рис. 13).

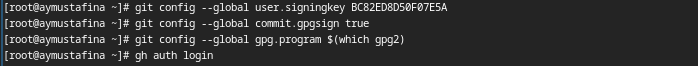


Рис. 13: Настройка подписей git

## 3.8 Настройка gh

Сначала аворизуюсь в gh, отвечаю на наводящие вопросы, в конце выбираю авторизацию через браузер (рис. 14).

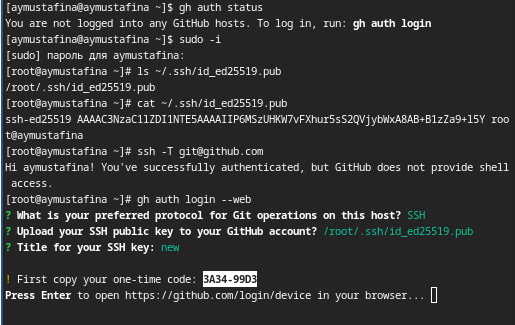


Рис. 14: Автoризация в gh

Завершаю авторизацию на сайте (рис. 15).

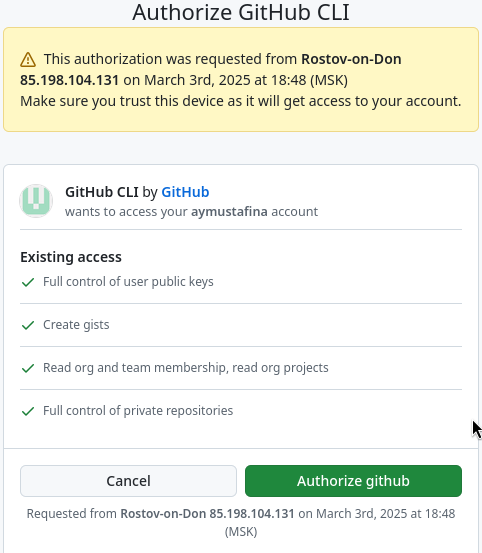


Рис. 15: Авторизация на сайте

Вижу сообщение о завершении авторизации на сайте (рис. 16).

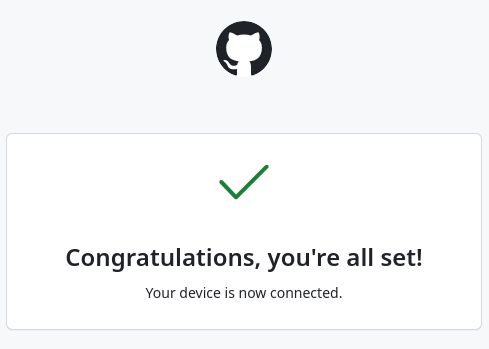


Рис. 16: Завершение авторизации

Вижу сообщение о завершении авторизации в терминале (рис. 17).

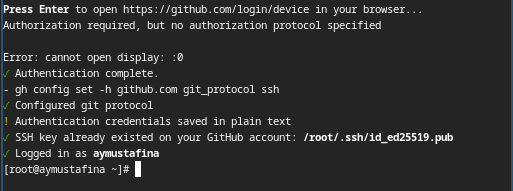


Рис. 17: Завершение авторизации в терминале

## 3.9 Создание репозитория курса на основе шаблона

Сначала создаю директорию с помощью утилиты mkdir и с помощью cd перехожу в созданную директорию:

mkdir -p ~/work/study/2024-2025/“Операционные системы” cd ~/work/study/2024-2025/“Операционные системы” gh repo create study\_2024-2025\_os-intro –template=yamadharma/course-directory-student-template –public git clone –recursive git@github.com:/study\_2024-2025\_os-intro.git os-intro (рис. 18).

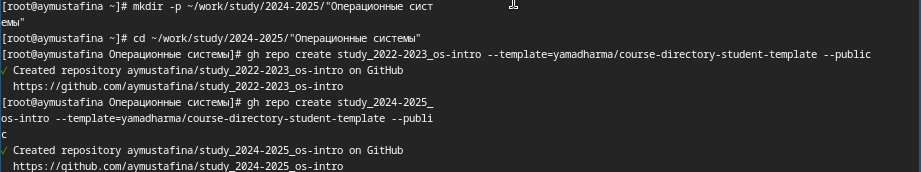


Рис. 18: Создание репозитория курса

Клонирование репозитория (рис. 19).

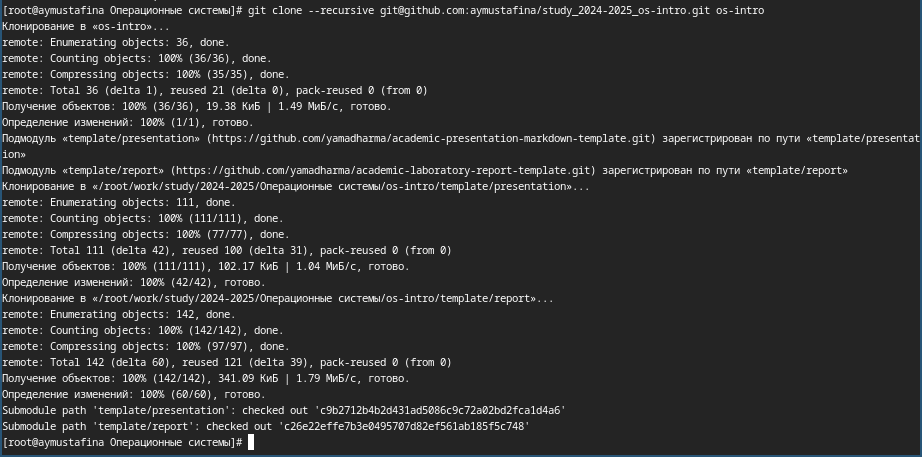


Рис. 19: Клонирование репозитория курса

Перехожу в каталог курса с помощью cd. Удалаю лишние файлы rm package.json. Создаю необходимые каталоги: echo os-intro > COURSE и make (рис. 20).

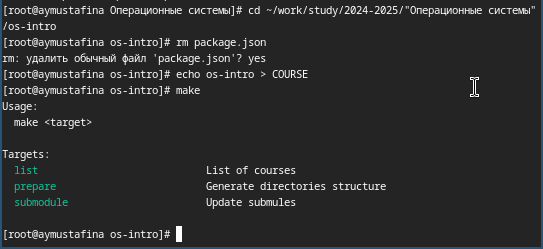


Рис. 20: Редактирование созданного репозитория

Добавляю все новые файлы для отправки на сервер и комментирую их: git add . git commit -am ‘feat(main): make course structure’ (рис. 21).

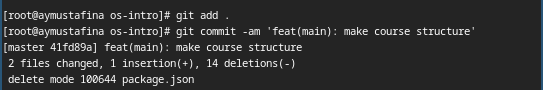


Рис. 21: Отправка файлов на сервер

Отправляю файлы на сервер с помощью git push (рис. 22).

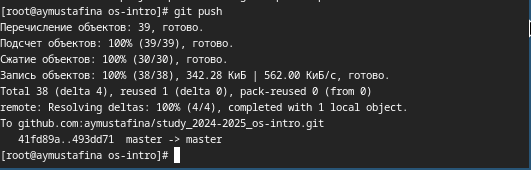


Рис. 22: Отправка файлов

# 4 Выводы

При выполнении данной лабораторной работы я изучила идеалогию и применение средств контроля версий, изучила команды для работы с git.

# 5 Ответы на контрольные вопросы

* Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

Системы контроля версий(VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Они помогают разработчикам отслеживать изменения и управлять версиями кода. Основная цель VCS - упростить и упорядочить работу над проектом. С его помощью можно легко отслеживать, какие изменения были внесены в проект и кем. Это помогает быстро находить и испрвалять ошибки, анализировать причины их возникновения.

* Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

Хранилище - это репозиторий, хранилице версий. В нем хранаятся документы и их история изменения. Commit - отслеживание редактирования файла и его изменений. Сохранение разницы между версиями. История - хранение всех изменений в проекте, благодаря чему возможно обратиться к данным, которые были изменены. Рабочая копия - копия проекта, которая которая используется в данный момент. Обычно последняя измененнная версия копия.

* Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

Централизованные системы контроля версий преполагают хранение проекта на едином сервере. Так чтобы изменить исходный код, разработчику необходимо скачать с сервера необходимые файлы и изменить, а затем снова вернуть эти файлы на сервер. Примеры централизованных VCS: CVS, Subversion, Perforce.

Децентрализованные системы контроля версий препологают, что при каждом копировании удаленного репазитория происходит полное копирование данных в локальный репозиторий. И каждая новая копия хранит все данные, хранящиеся в удаленном репазитории. Примеры децентрализованных VCS: Git, Basaar, Mercorial.

* Опишите действия с VCS при единоличной работе с хранилищем.

Создается и подключается удаленный репазиторий. Далее при изменении поекта новые изменения отправляютя на сервер.

* Опишите порядок работы с общим хранилищем VCS.

Сначала разработчик клонирует нужный ему удаленный репазиторий. Создает ветку для работы. После внесения каких-либо изменений в код, разработчик отправляет изменения в удаленный репазиторий. ПРи этом все внесенные изменения сохраняются и к ним можно вернуться в любой момент. - Каковы основные задачи, решаемые инструментальным средством git?

управление версиями файлов отслеживание изменений резервное копирование и восстановление данных

* Назовите и дайте краткую характеристику командам git.

Создание основного дерева репозитория:

git init

Получение обновлений (изменений) текущего дерева из центрального репозитория:

git pull

Отправка всех произведённых изменений локального дерева в центральный репозиторий:

git push

Просмотр списка изменённых файлов в текущей директории:

git status

Просмотр текущих изменений:

git diff

Сохранение текущих изменений:

добавить все изменённые и/или созданные файлы и/или каталоги:

git add .

добавить конкретные изменённые и/или созданные файлы и/или каталоги:

git add имена\_файлов

удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории):

git rm имена\_файлов Сохранение добавленных изменений:

сохранить все добавленные изменения и все изменённые файлы:

git commit -am ‘Описание коммита’

сохранить добавленные изменения с внесением комментария через встроенный редактор:

git commit

создание новой ветки, базирующейся на текущей:

git checkout -b имя\_ветки

переключение на некоторую ветку:

git checkout имя\_ветки

(при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)

отправка изменений конкретной ветки в центральный репозиторий:

git push origin имя\_ветки

слияние ветки с текущим деревом:

git merge –no-ff имя\_ветки

Удаление ветки: удаление локальной уже слитой с основным деревом ветки:

git branch -d имя\_ветки

принудительное удаление локальной ветки:

git branch -D имя\_ветки

удаление ветки с центрального репозитория:

git push origin :имя\_ветки

* Приведите примеры использования при работе с локальным и удалённым репозиториями.

Для локального репозитория: git init - создание новго репозитория git add . - добавление всех файлов в индекс git commit -m “Initial commit” - фиксируем изменения

Для удаленного репозитория: git clone - клонирование репозитория git push origin main - отправка изменений в удаленный репозиторий git pull - плоучение изменений из удаленного репозитория

* Что такое и зачем могут быть нужны ветви (branches)?

Отдельные линии разработки, позволяющие работать над разными задачами независимо. Нужны для изоляции изменений, тестирования, испрваления багов без влияния на основную ветку.

* Как и зачем можно игнорировать некоторые файлы при commit?

Создается файл .gitignore, в котором указываются шаблоны файлов/папок, которые следует игнорировать. Это делается для того, чтобы не включать в репозиторий временные файлы, бинарные данные =, конфиденциальную информацию, файлы, создаваемые средой разработки.

# 6 Список литературы

1. Лабораторная работа №2 [1]. URL: <https://esystem.rudn.ru/mod/page/view.php?id=1224371>

1. [Лабораторная работа №2](https://esystem.rudn.ru/mod/page/view.php?id=1224371).