Лабораторная работа №2

Операционные системы

Краснова Камилла Геннадьевна

Содержание

# 1 Цель работы

Целью данной лабораторной работы является изучение идеологии и применения средств контроля версий, а также освоение умений по работе с git.

# 2 Задание

1. Создать базовую конфигурацию для работы с git
2. Создать ключ SSH
3. Создать ключ GPG
4. Настроить подписи Git
5. Заргеистрироваться на GitHub
6. Создать локальный каталог для выполнения заданий по предмету.

# 3 Выполнение лабораторной работы

## 3.1 Установка программного обеспечения

Захожу в терминал и устанавливаю git, следом устанавливаю gh (рис. 1).

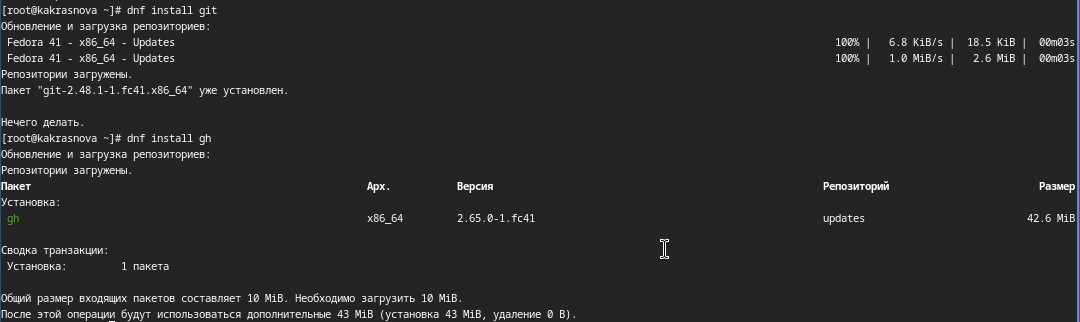


Рис. 1: Установка git и gh

## 3.2 Базовая настройка git

Задаю имя и email владельца репозитория (рис. 2).

Рис. 2: Данные владельца

Рис. 2: Данные владельца

Настраиваю utf-8 в выводе сообщений git (рис. 3).

Рис. 3: Настройка utf-8

Рис. 3: Настройка utf-8

Задаю имя начальной ветки (рис. 4).

Рис. 4: Имя начальной ветки

Рис. 4: Имя начальной ветки

Далее задаю параметры autocrlf и safecrlf (рис. 5).

Рис. 5: Параметры autocrlf и safecrlf

Рис. 5: Параметры autocrlf и safecrlf

## 3.3 Создание ключей ssh

Создаю ключ размером 4096 бит по алгоритму rsa (рис. 6).

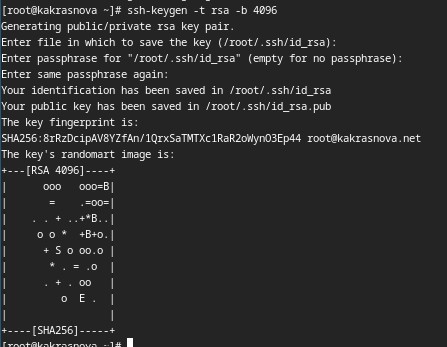


Рис. 6: Алгоритм rsa

Дальше создаю ключ по алгоритму ed25519 (рис. 7).

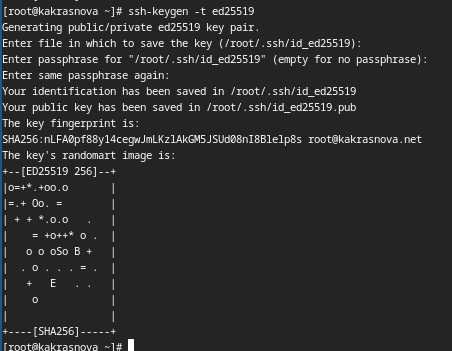


Рис. 7: Алгоритм ed25519

## 3.4 Создание ключей gpg

Генерирую ключ (рис. 8).

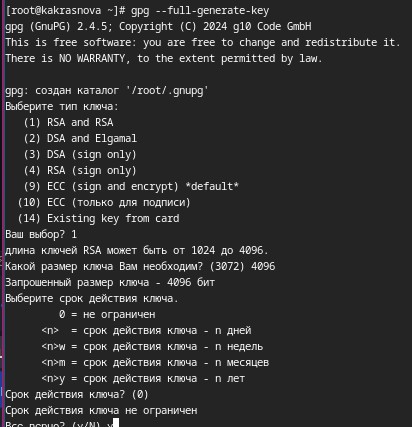


Рис. 8: Генерация ключа

И выбираю нужные опции. Подтверждаю и придумываю кодовую фразу (рис. 9).

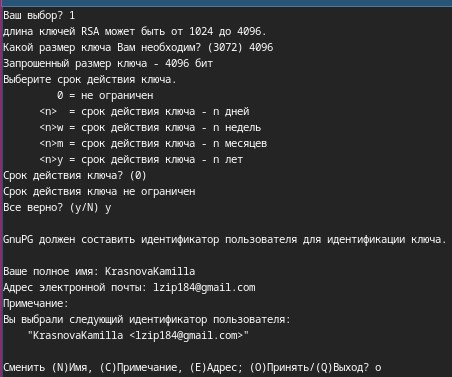


Рис. 9: Генерация ключа

## 3.5 Настройка github

Учетная запись у меня уже была создана, а также были заполнены основные данные (рис. 10).

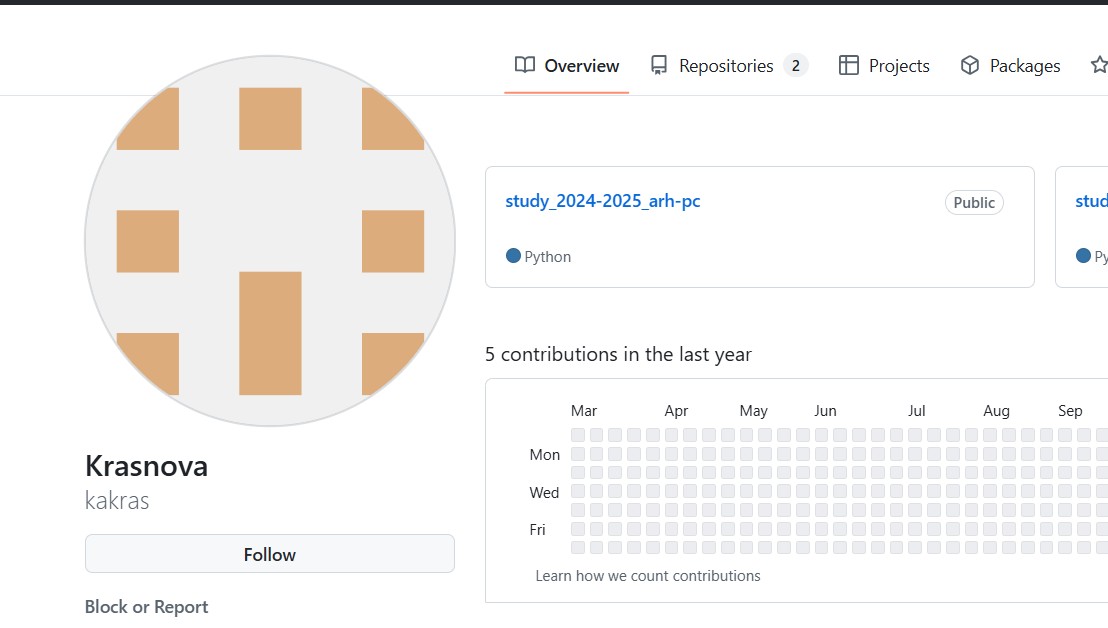


Рис. 10: Учетная запись github

## 3.6 Добавление gpg ключа в Github

Вывожу список ключей и копирую отпечаток приватного ключа (рис. 11).

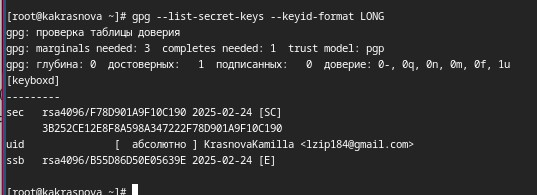


Рис. 11: Список ключей

Копирую сгенерированный gpg ключ в буфер обменя и вставляю полученнный ключ в поле ввода в настройках GitHub (рис. 12).

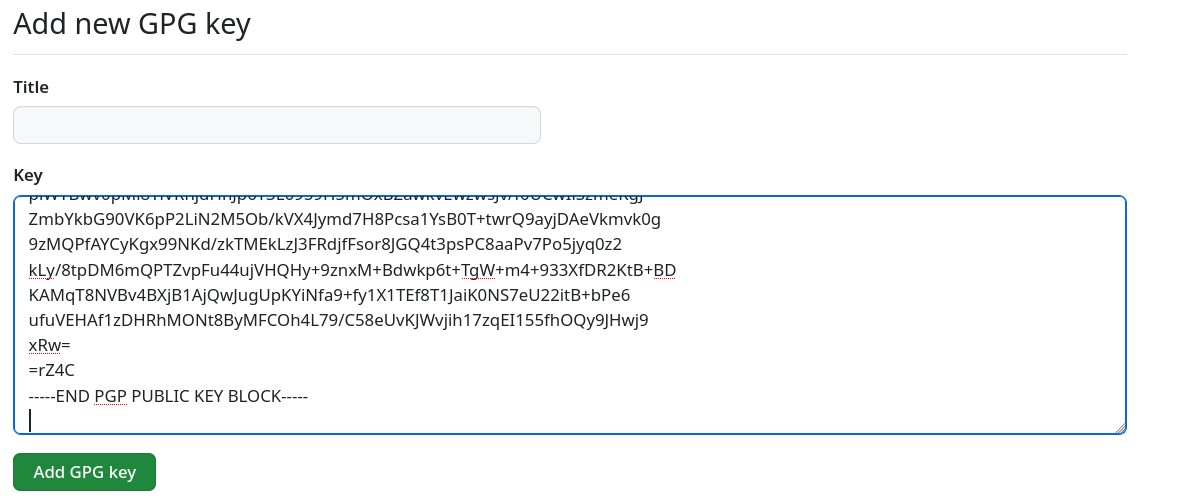


Рис. 12: New GPG key

GPG ключ добавился в GitHub (рис. 13).

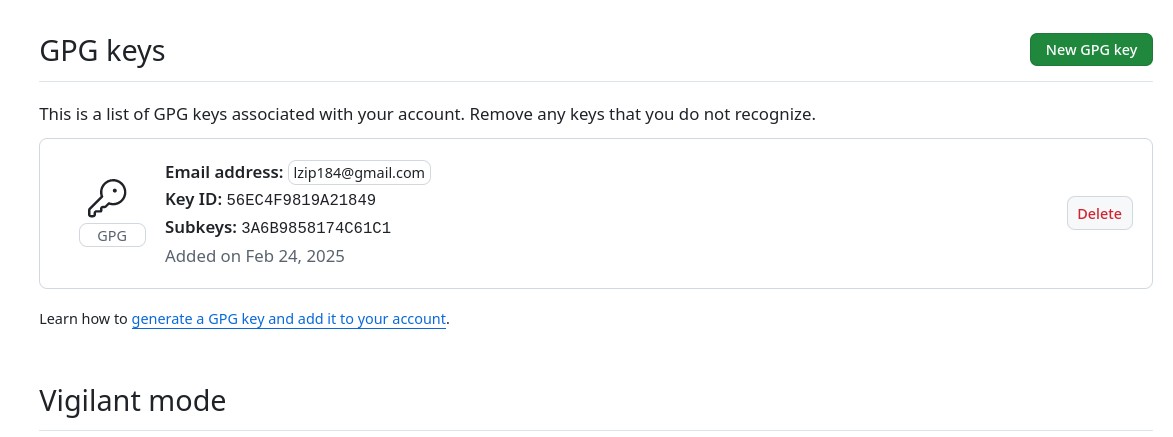


Рис. 13: Готовый gpg ключ

## 3.7 Настройка автоматических подписей коммитов git

Использую введеный email, указываю Git применять его при подписи коммитов (рис. 14).

Рис. 14: Автоматические подписи

Рис. 14: Автоматические подписи

## 3.8 Настройка gh

Авторизуюсь и отвечаю на несколько наводящих вопросов от утилиты. В конце выбираю авторизацию через токен. Создаю токен на GitHub и вставляю его в терминал. Получаю сообщение об успешной авторизации (рис. 15).

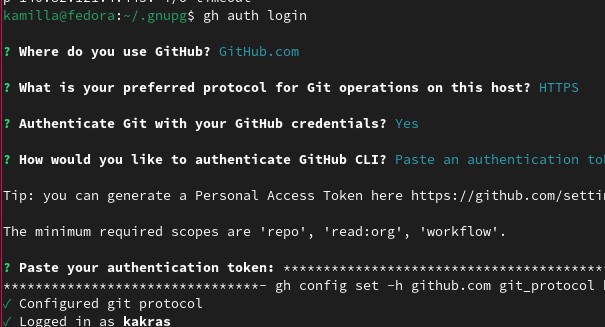


Рис. 15: Авторизация

## 3.9 Создание репозитория курса на основе шаблона

У меня ууже был создан репозиторий курса (рис. 16).

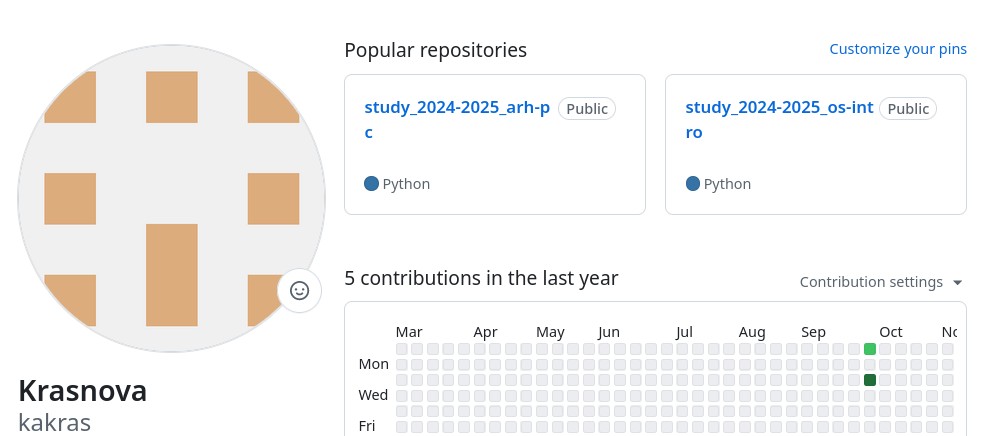


Рис. 16: Репозиторий курса

## 3.10 Настройка каталога курса

Также у меня уже был настроен каталог курса (рис. 17).

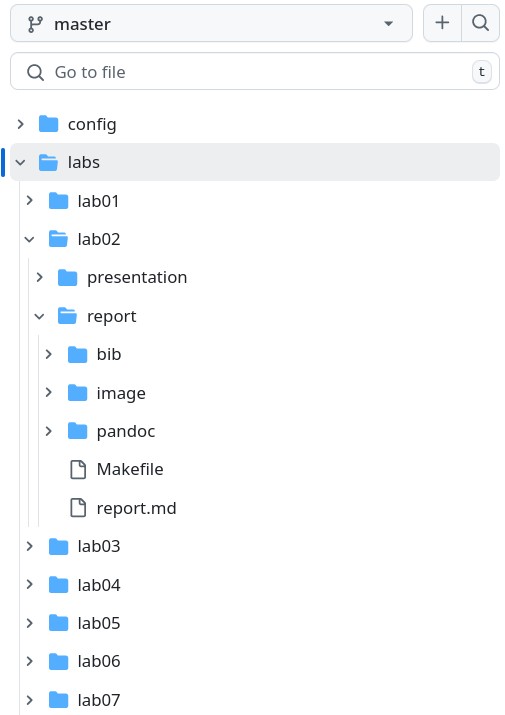


Рис. 17: Каталог курса

# 4 Выводы

При выполнении данной лабораторной работы я изучила идеологию и применение средств контроля версий, а также освоила умения по работе с git.

# 5 Ответы на контрольные вопросы.

1. Системы контроля версий (VCS) - программное обеспечение для облегчения работы с изменяющейся информацией. Они позволяют хранить несколько версий изменяющейся информации, одного и того же документа, может предоставить доступ к более ранним версиям документа. Используется для работы нескольких человек над проектом, позволяет посмотреть, кто и когда внес какое-либо изменение и т. д. VCS ррименяются для: Хранения понлой истории изменений, сохранения причин всех изменений, поиска причин изменений и совершивших изменение, совместной работы над проектами.
2. Хранилище – репозиторий, хранилище версий, в нем хранятся все документы, включая историю их изменения и прочей служебной информацией. commit – отслеживание изменений, сохраняет разницу в изменениях. История – хранит все изменения в проекте и позволяет при необходимости вернуться/обратиться к нужным данным. Рабочая копия – копия проекта, основанная на версии из хранилища, чаще всего последней версии.
3. Централизованные VCS (например: CVS, TFS, AccuRev) – одно основное хранилище всего проекта. Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет, затем добавляет изменения обратно в хранилище. Децентрализованные VCS (например: Git, Bazaar) – у каждого пользователя свой вариант репозитория (возможно несколько вариантов), есть возможность добавлять и забирать изменения из любого репозитория. В отличие от классических, в распределенных (децентралиованных) системах контроля версий центральный репозиторий не является обязательным.
4. Сначала создается и подключается удаленный репозиторий, затем по мере изменения проекта эти изменения отправляются на сервер.
5. Участник проекта перед началом работы получает нужную ему версию проекта в хранилище, с помощью определенных команд, после внесения изменений пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются. К ним можно вернуться в любой момент.
6. Хранение информации о всех изменениях в вашем коде, обеспечение удобства командной работы над кодом.
7. 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

1. git push -all отправляем из локального репозитория все сохраненные изменения в центральный репозиторий, предварительно создав локальный репозиторий и сделав предварительную конфигурацию.
2. Ветвление - один из параллельных участков в одном хранилище, исходящих из одной версии, обычно есть главная ветка. Между ветками, т. е. их концами возможно их слияние. Используются для разработки новых функций.
3. Во время работы над проектом могут создаваться файлы, которые не следуют добавлять в репозиторий. Например, временные файлы. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов.

# Список литературы

1. Лабораторная работа № 2 [Электронный ресурс] URL: https://esystem.rudn.ru/mod/page/view.php?id=970819