Отчет по лабораторной работе №2

Операционные системы

Самарханова Полина Тимуровна

Содержание

# 1 Цель работы

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

# 2 Задания

1. Создание базовой конфигурации для работы с git
2. Создание SSH ключа
3. Создание GPG ключа
4. Настройка подписей git
5. Регистрация на Github
6. Создание локального каталога для выполнения заданий по предмету

# 3 Выполнение лабораторной работы

## 3.1 Установка ПО

Устанавливаю необходимое ПО git и gh через терминал с помощью команд dnf install git, dnf install gh

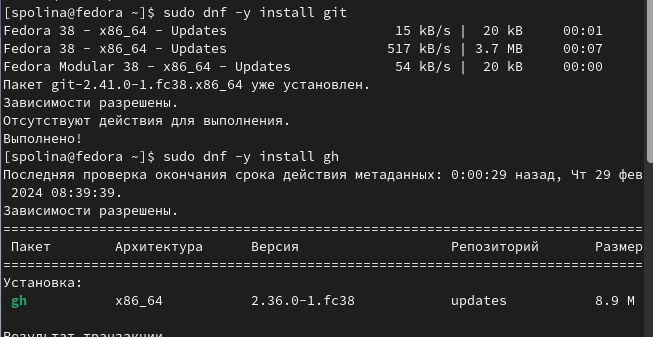


Рис. 3.1 Установка git и gh

## 3.2 Базовая настройка git

Задаю в качестве имени и почты владельца свои имя с фамилией и почту соответственно

Рис. 3.2 Задаю имя и email владельца репозитория

Рис. 3.2 Задаю имя и email владельца репозитория

Настраиваю utf-8 в выводе сообщений git для их корректного отображения

Рис. 3.3 Настройка utf-8 в выводе сообщений git

Рис. 3.3 Настройка utf-8 в выводе сообщений git

Начальной ветке задаю имя master

Рис. 3.4 Задаю имя начальной ветки

Рис. 3.4 Задаю имя начальной ветки

Задаю параметры autocrlf и safecrlf для корректного отображения конца строки

Рис. 3.5 Задаю параметры autocrlf и safecrlf

Рис. 3.5 Задаю параметры autocrlf и safecrlf

## 3.3 Создание ключа SSH

Создаю ключ SSH размером 4096 бит по алгоритму rsa

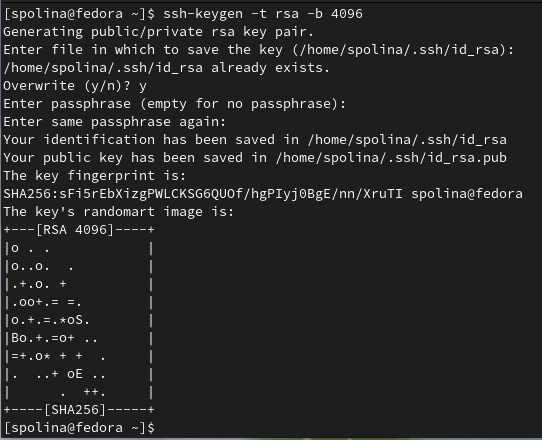


Рис. 3.6 Генерация SSH ключа по алгоритму rsa

Создаю ключ SSH по алгоритму ed25519

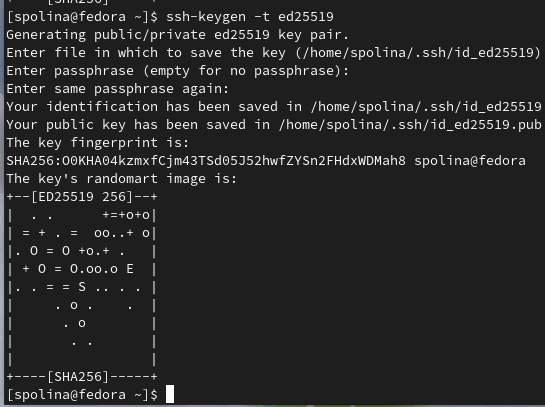


Рис. 3.7 Генерация SSH ключа по алгоритму ed25519

## 3.4 Создание ключа GPG

Генерирую ключ GPG, затем выбираю тип ключа RSA и RSA, длина ключа 4096 бит, срок действия - неограниченный. Затем отвечаю на запрашиваемую личную информацию

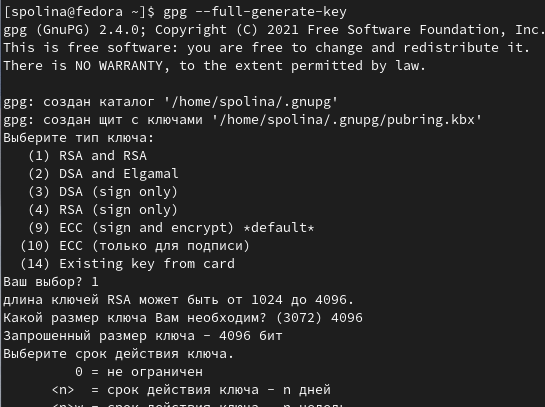


Рис. 3.8 Генерация GPG ключа

## 3.5 Регистрация на GitHub

Аккаунт у меня уже был, поэтому просто вхожу в него

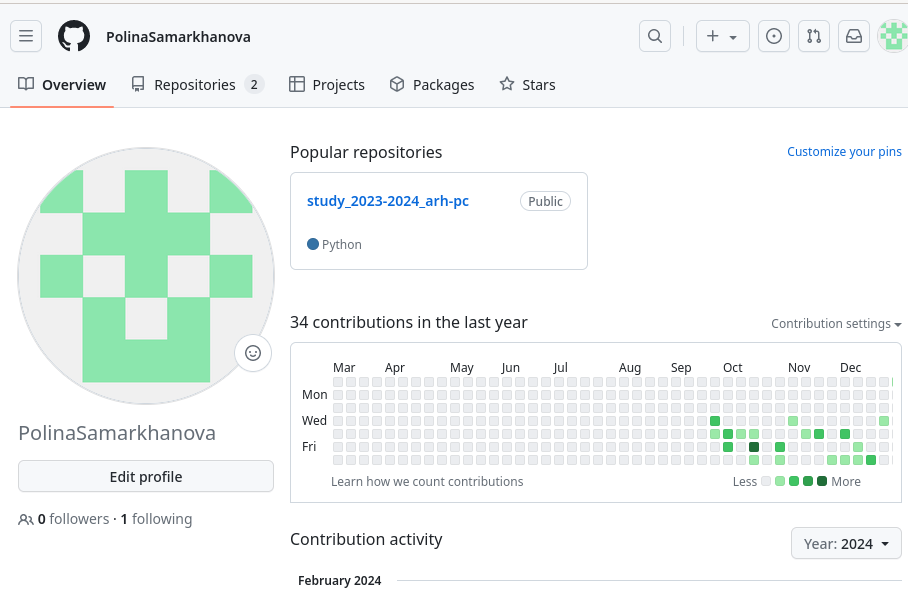


Рис. 3.9 Аккаунт на GitHub

## 3.6 Добавление ключа GPG в Github

Выводим список ключей и копируем в буфер обмена отпечаток приватного ключа (последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа), который находится после знака слеша

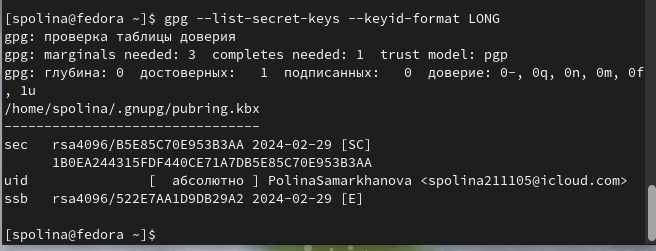


Рис. 3.10 Вывод списка ключей

Ввожу в терминале команду gpg –armor –export | xclip -sel clip С ее помощью копирую сам ключ в буфер обмена

Рис. 3.11 Копирование ключа в буфер обмена

Рис. 3.11 Копирование ключа в буфер обмена

Далее открываю настройки Github и ищу раздел GPG ключей

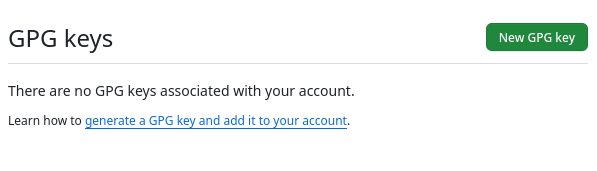


Рис. 3.12 Настройка Github

Нажимаю кнопку New GPG keyи вставляю в нужное поле скопированный ключ

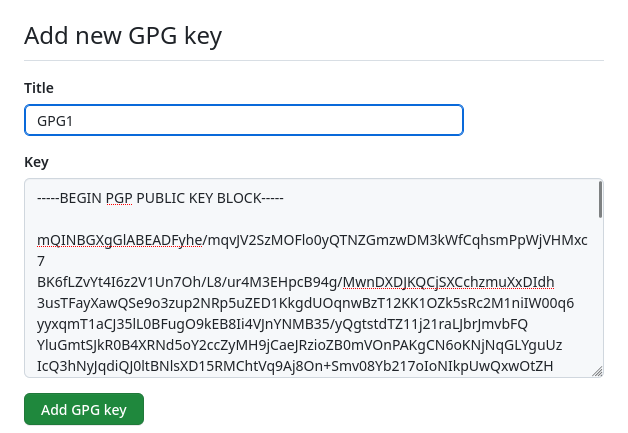


Рис. 3.13 Добавление ключа

Теперь можно увидеть добавленный на Github ключ GPG

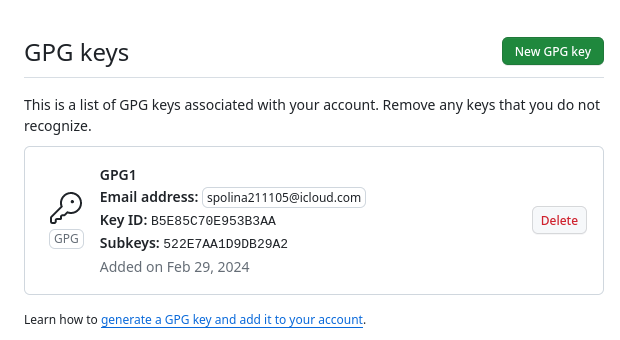


Рис. 3.14 Добавленный ключ GPG

Далее настраиваю автоматические подписи коммитов Git

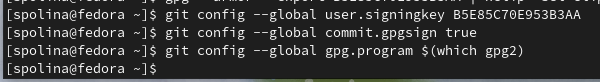


Рис. 3.15 Настройка подписей Git

## 3.7 Настройка gh

Начинаю с авторизации в gh: нужно было ответить на вопросы утилиты и в конце выбрать авторизацию через браузер.

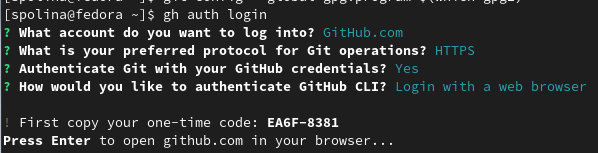


Рис. 3.16 Авторизация в gh

Скопировав 8-значный код и вставив его в открывшемся окне браузера, завершаю авторизацию

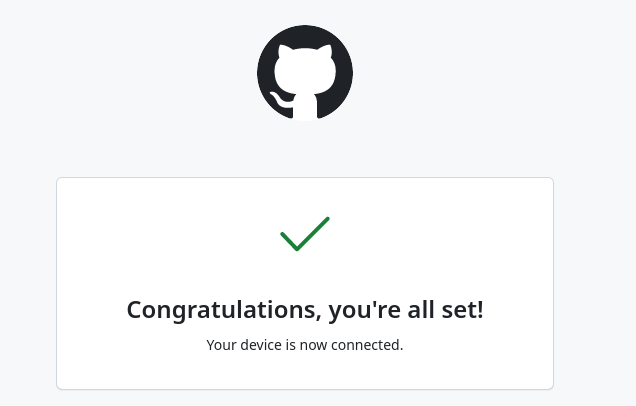


Рис. 3.17 Завершение авторизации через браузер

Теперь можно увидеть завершение авторизации и в терминале

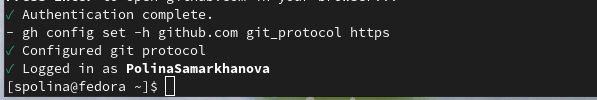


Рис. 3.18 Завершение авторизации

## 3.8 Сознание репозитория курса на основе шаблона

Сначала я создала новую папку с помощью утилиты mkdir: mkdir -p ~/work/study/2023-2024/“Операционные системы” Далее я перешла в саму папку Операционные системы через утилиту cd Затем я в терминале ввела команду: gh repo create study\_2023-2024\_os-intro –template=yamadharma/course-directory-student-template –public Она нужна для того, чтобы создать репозиторий на основе указанного шаблона после этого я клонировала репозиторий к себе в папку, но указала ссылку с протоколом hhtps, а не ssh

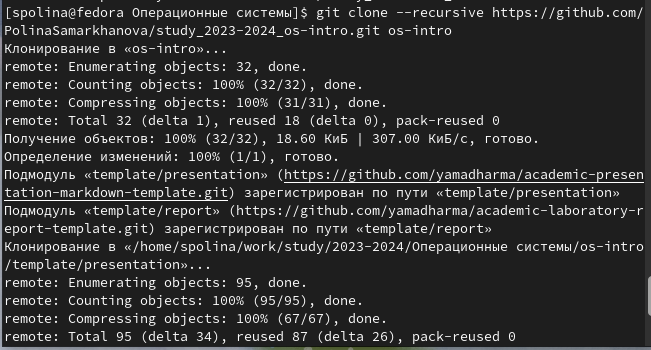


Рис. 3.19 Создание рпеозитория

Перехожу в каталог курса с помощью утилиты cd, затем с помощью ls проверяю содержимое каталога

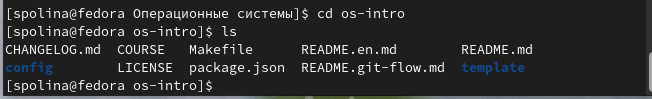


Рис. 3.20 Перемещение в нужную директорию

Удаляю лишние файлы с помощью команды rm, затем создаю необходимые каталоги, используя makefile, а именно: make list make prepare make submodule

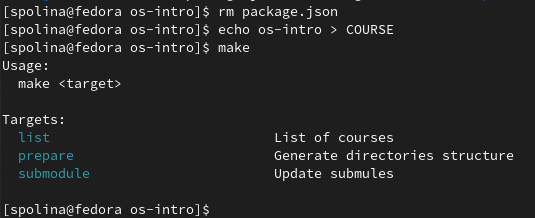


Рис. 3.21 Удаление файлов и создание каталогов

Добавляю все новые файлы для отправки на сервер с помощью git add и комментирую при помощи git commit

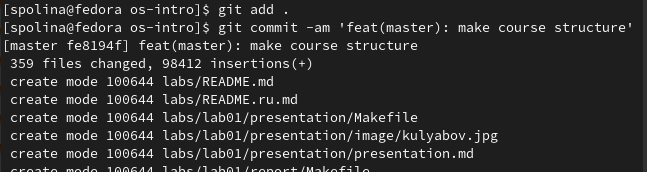


Рис. 3.22 Отправка файлов на сервер

Отправляю файлы на сервер с помощью git push

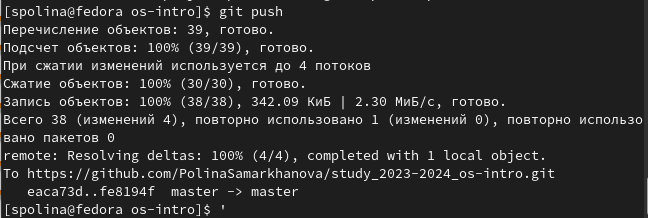


Рис. 3.23 Отправка файлов на сервер

# 4 Выводы

При выполнении данной лабораторной работы я идеологию и применение средств контроля версий, а также освоила навыки 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=1098790