Лабораторная работа №2.

Первоначальная настройка git.

Кучеренко София

Содержание

# 1 Цель работы

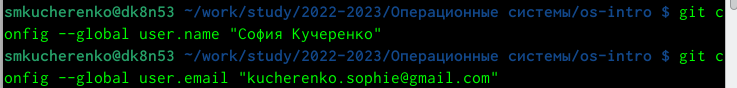
1. Изучить идеологию и применение средств контроля версий.
2. Освоить умения по работе с git.

# 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

Установим git и зададим имя и email владельца репозитория:



Имя и email пользователя

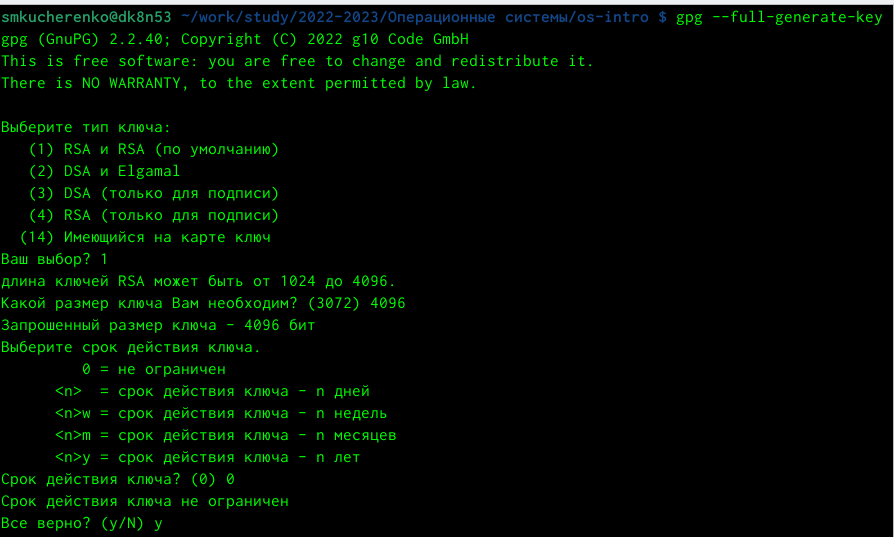
Настроим utf-8 в выводе сообщений git:

Настройка utf-8

Настройка utf-8

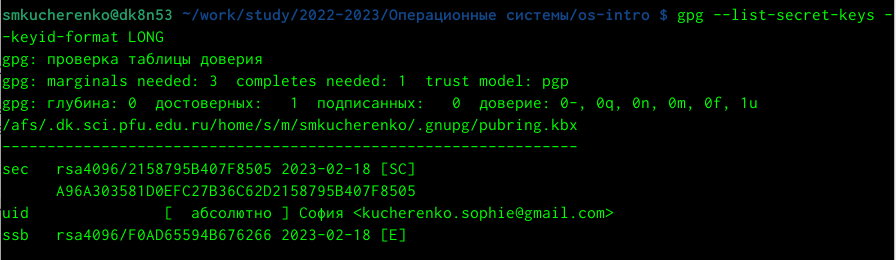
Настройте верификацию и подписание коммитов git:

Генерируем ключ и из предложенных опций выбираем: тип RSA and RSA; размер 4096; срок действия 0 (не истекает никогда), указываем имя и email.



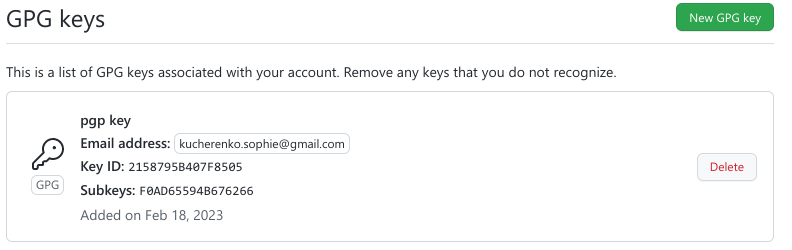
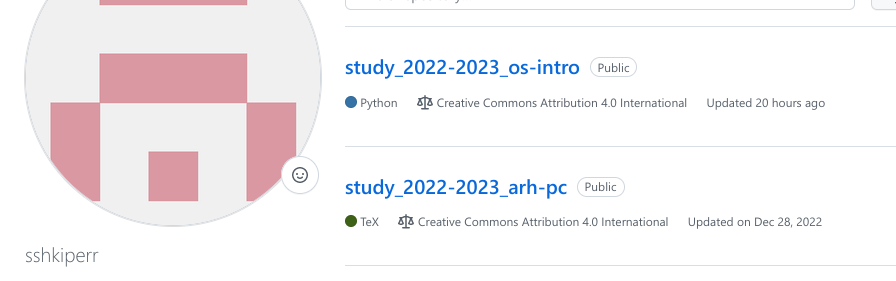
Создание ключа

Выводим список ключей и копируем отпечаток приватного ключа:

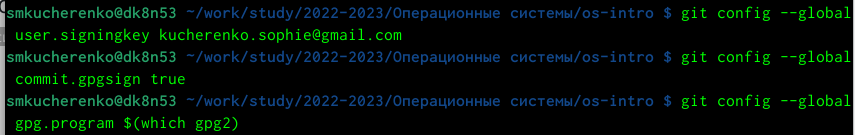


Список ключей

Копируем ключ и добавляем его в настройках профиля на GitHub, предварительно авторизоваашись:

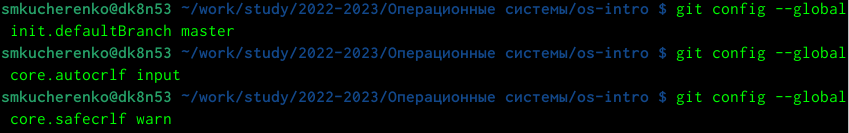
 

Используя введёный email, укажем Git применять его при подписи коммитов:



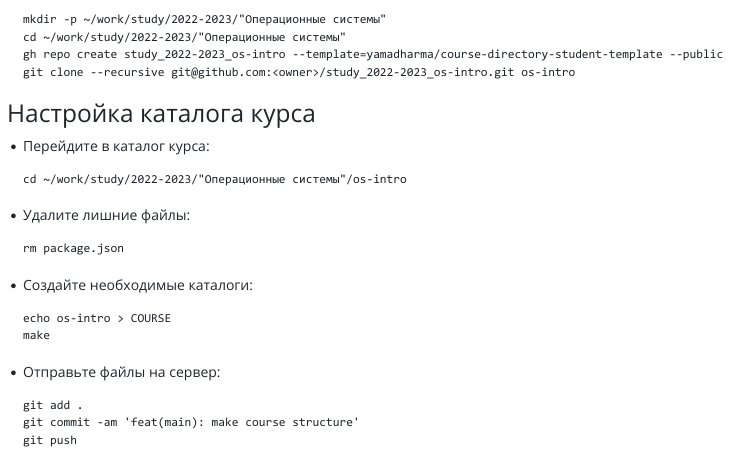
Название рисунка

Зададим имя начальной ветки и укажем некоторые параметры:



Название рисунка

Также перед началом выполнения рабораторных работ я создала рабочее пространство, выполнив ряд действий:



Создание рабочего пространства

# 4 Контрольные вопросы

1. 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.
2. 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. Изменив содержимое рабочей копии, разработчик фиксирует (*commit*) сделанные изменения в репозитории. Как правило, фиксация сопровождается небольшим текстовым комментарием.

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.

Для работы с содержимым репозитория каждый разработчик имеет собственную *рабочую копию*. Рабочая копия- «снимок» содержимого репозитория и некоторая служебная информация.

1. В *централизованной* системе все пользователи подключены к центральному владельцу сети или «серверу». Центральный владелец хранит данные, к которым могут получить доступ другие пользователи, а также информацию о пользователях. Эта информация о пользователе может включать профили пользователей, пользовательский контент и многое другое. Централизованная система проста в установке и может быстро развиваться. Но у этой системы есть важное ограничение. Если сервер выходит из строя, система перестает работать должным образом, и пользователи не могут получить доступ к данным. (CVS, Subversion)

У *децентрализованных* систем нет единого центрального владельца. Вместо этого они используют нескольких центральных владельцев, каждый из которых обычно хранит копию ресурсов, к которым пользователи могут получить доступ. Децентрализованная система может быть так же уязвима к сбоям, как и централизованная. Однако по своей конструкции система более устойчива к неисправностям. (Git, Mercurial)

1. При единоличной работе с хранилищем все изменения, созданные пользователем, не влияют на общий репозиторий.
2. Опишите порядок работы с общим хранилищем VCS: Из общего хранилища можно получать изменения проекта.
3. Каковы основные задачи, решаемые инструментальным средством git? git позволяет несольким людям работать над одним проектом.
4. add - добавить файлы в коммит, push - отправить коммит на удалённый репозиторий; pull - импортировать проект с удалённого репозитория.
5. Приведите примеры использования при работе с локальным и удалённым репозиториями.

Ответ:

1. Что такое и зачем могут быть нужны ветви (branches)? Создав новую ветвь, можно, не вредя проекту, работать над конкретной частью проекта.
2. Как и зачем можно игнорировать некоторые файлы при commit? some files may well be user specific. gitignore нужен для скрытия файлов и папок от системы контроля версий Git. Обычно скрывают конфигурационные файлы (особенно с паролями), временные файли и папки.

# 5 Выводы

В результате выполнения лабораторной работы была изучена идеология средств контроля версий, а также были приобретены навыки по работе с git.