Отчёта по лабораторной работе №2

Управление версиями

Смирнов-Мальцев Егор Дмитриевич

# Цель работы

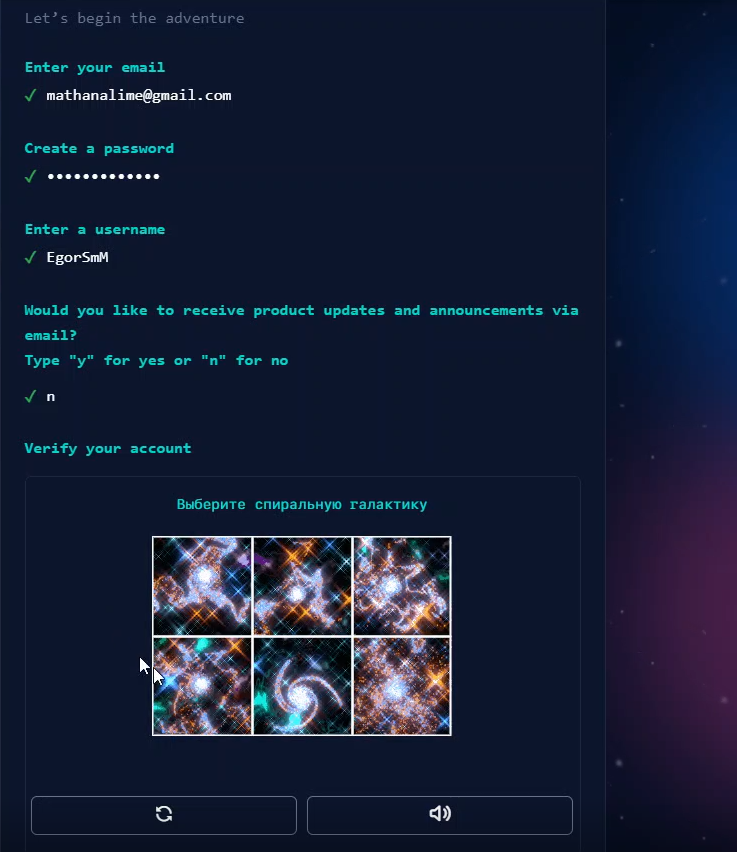
* Изучить идеологию и применение средств контроля версий.
* Освоить умения по работе с git.

# Задание

* Создать базовую конфигурацию для работы с git.
* Создать ключ SSH.
* Создать ключ PGP.
* Настроить подписи git.
* Зарегистрироваться на Github.
* Создать локальный каталог для выполнения заданий по предмету.

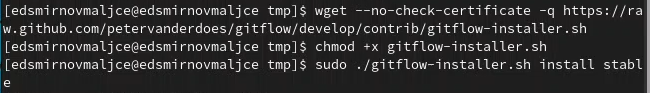
# Выполнение лабораторной работы

1. Создал учетную запись на Github и заполнил основные данные.



Страница регистрации на Github

1. Установил Git-flow и gh на виртуальной машине.

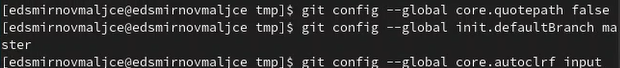


Установка git-flow

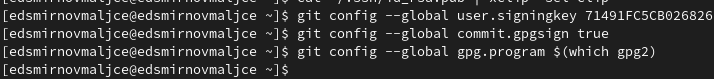
mountains

Установка gh

1. Настроил git. (рис. [-@fig:004], [-@fig:005])

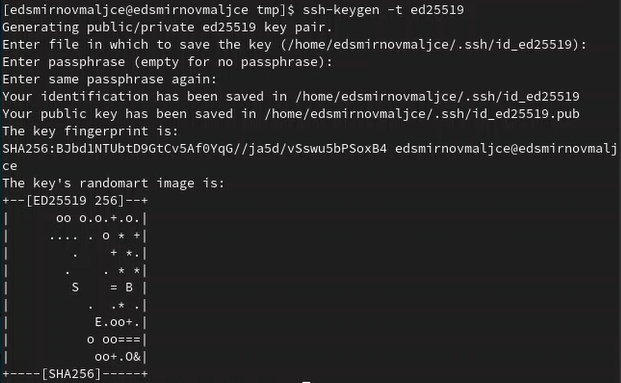


Создание основной ветки

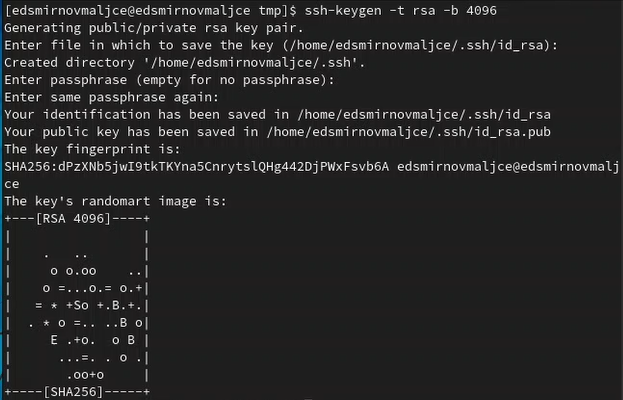


Унифицирование коммитов

1. Создал 2 ssh ключа: по алгоритму rsa размером 4096 бит и по алгоритму ed25519.

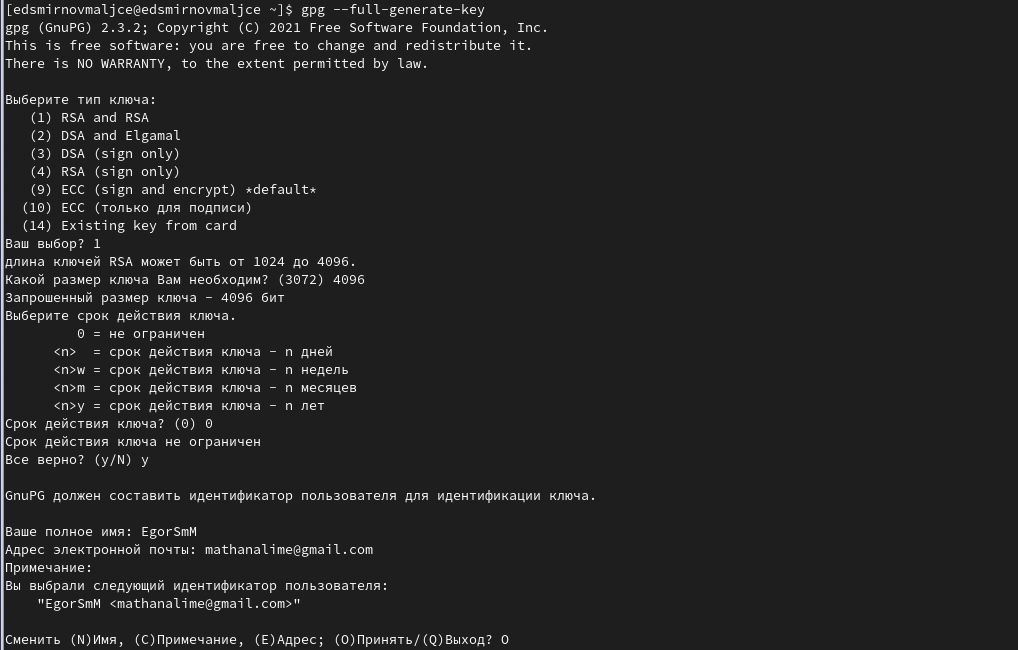


Создание ключа по алгоритму ed25519



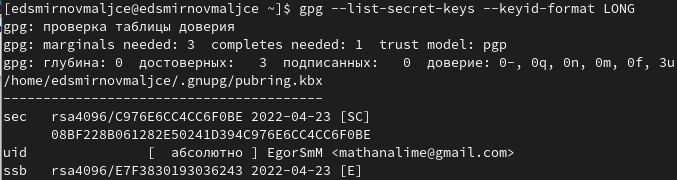
Создание ключа по алгоритму rsa размером 4096 бит

1. Сгенерировал pgp-ключ.



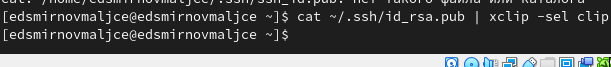
Генерация pgp-ключа

1. Скопировал pgp-ключ и ввел его на сайте Github.



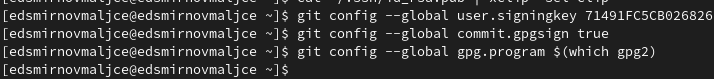
Копирование pgp-ключа

1. Скопировал ssh-ключ и ввел его на Github.



Копирование ssh-ключа

1. Указал Git применять ранее введенный email в качестве подписи коммитов.



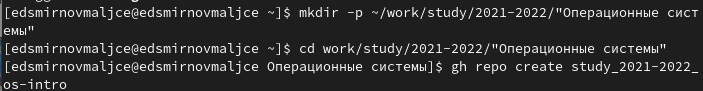
Настройка автоматических подписей

1. Авторизировался на Github.

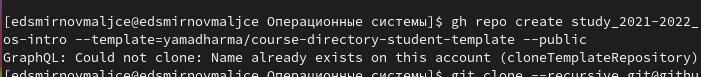


Переход к авторизации через командную строку

1. Создал репозиторий на основе шаблона рабочего пространства.



Создание директории и переход в нее

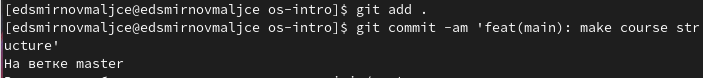


Копирование шаблона репозитория

1. Настроил репозиторий.

mountains

Удаление лишнего файла



Создание каталогов

mountains

Отправление файла на сервер

# Выводы

* Я научился создавать репозиторий на Github с помощью командной строки.
* Я ознакомился с Git структурой и понял ее преимущества при работе в команде, а именно удобство отслеживания изменений.

# Ответы на контрольные вопросы

1. VCS – это системы контроля версий. Они используются при работе в команде. 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. Хранилище – место, в котором хранятся все версии проекта. Для уменьшения объема используемой памяти можно хранить только изменения проекта. Commit – добавленные и измененные файлы по сравнению с предшествующей версией проекта. История – последовательность изменений проекта. Рабочая копия – копия над которой сейчас идет работа.
3. Централизованная система – система, в которой существует центральное хранилище, которое доступно всем участникам проекта. В децентрализованной системе у каждого участника есть свой репозиторий, что позволяет работать, не подключаясь к сети. Пример централизованной системы: CVS. Децентрализованной: Git.
4. При единоличной работе с VCS берешь нужную версию, вносишь правки и добавляешь коммит.
5. При совместной работе в централизованном хранилище, добавляется работа по устранению конфликтов. Также необходимо сливать версии, и вообще следить за структурированностью истории. Нельзя забывать проверять последнюю версию программы, потому что напарник мог ее изменить.
6. Git помогает работать одновременно над одним и тем же проектом независимо, а потом совмещать достижения разработчиков. Также он помогает хранить историю версий, что позволяет быстро откатиться в случае необходимости.
7. Основные команды 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 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 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: 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 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
8. Пример использования локального репозитория: В Git создать репозиторий привязанный к имени пользователя и email и работать в нем не подключаясь к сети. Пример использования удаленного репозитория: создать репозиторий на Github, настроить авторизацию через ssh-ключ и pgp-ключ, авторизоваться и работать с ним через командную строку.
9. Ветви необходимы в случае, если есть несколько путей развития программы, поэтому из одной версии получаются сразу несколько.
10. Игнорировать файлы можно с помощью команды gitignore для того, чтобы не забивать хранилище мусором.