Российский университет дружбы народов

Факультет физико-математических и естественных наук

**Лабораторная работа №2**

**Управление версиями**

**Отчет**

Выполнил: Смирнов-Мальцев Е.Д.

Преподаватель: Кулябов Д.С.

Москва

2022

**Цель работы**:

– Изучить идеологию и применение средств контроля версий.

– Освоить умения по работе с git.

**Ход работы**:

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

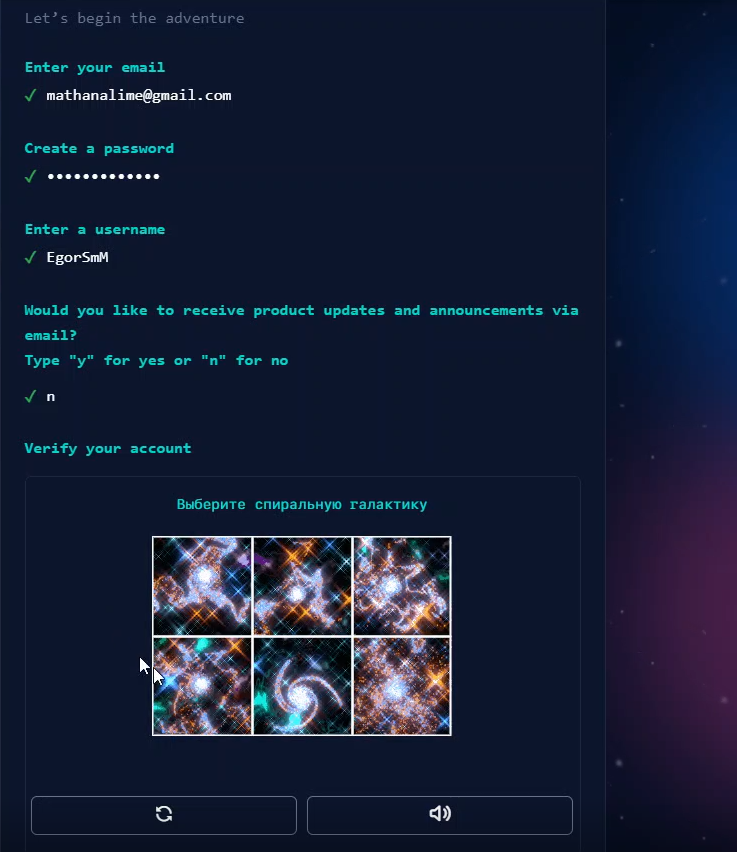


Рисунок Страница регистрации на Github

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

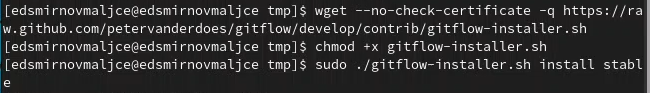


Рисунок Установка git-flow



Рисунок Установка gh

1. Настроил git

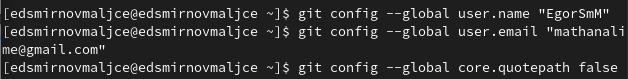


Рисунок Задание имени и email владельца репозитория. Настройка вывода в utf-8.

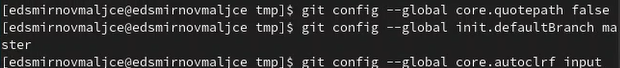


Рисунок Создание основной ветки

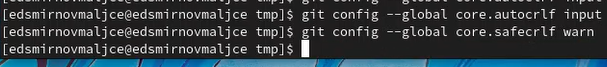


Рисунок Унифицирование коммитов

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

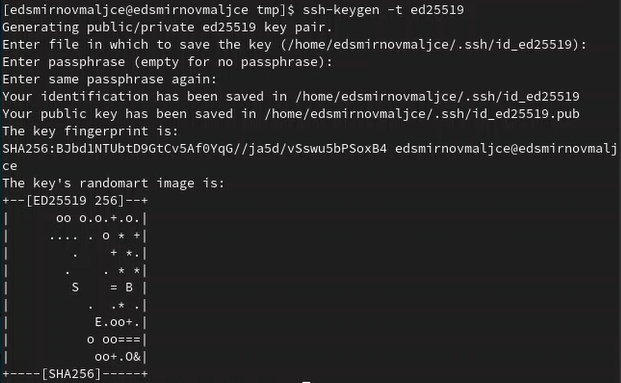


Рисунок Создание ключа по алгоритму ed25519

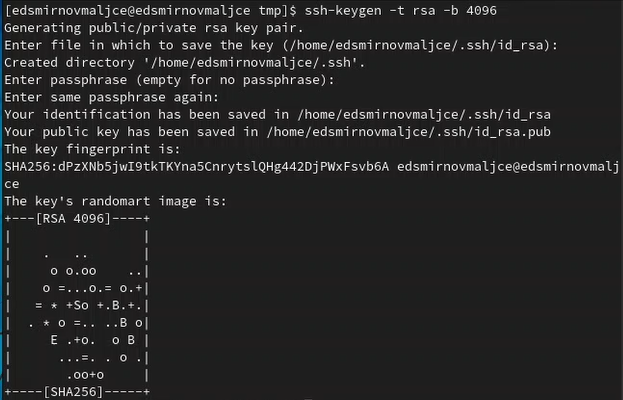


Рисунок Создание ключа по алгоритму rsa размером 4096 бит

1. Сгенерировал pgp-ключ

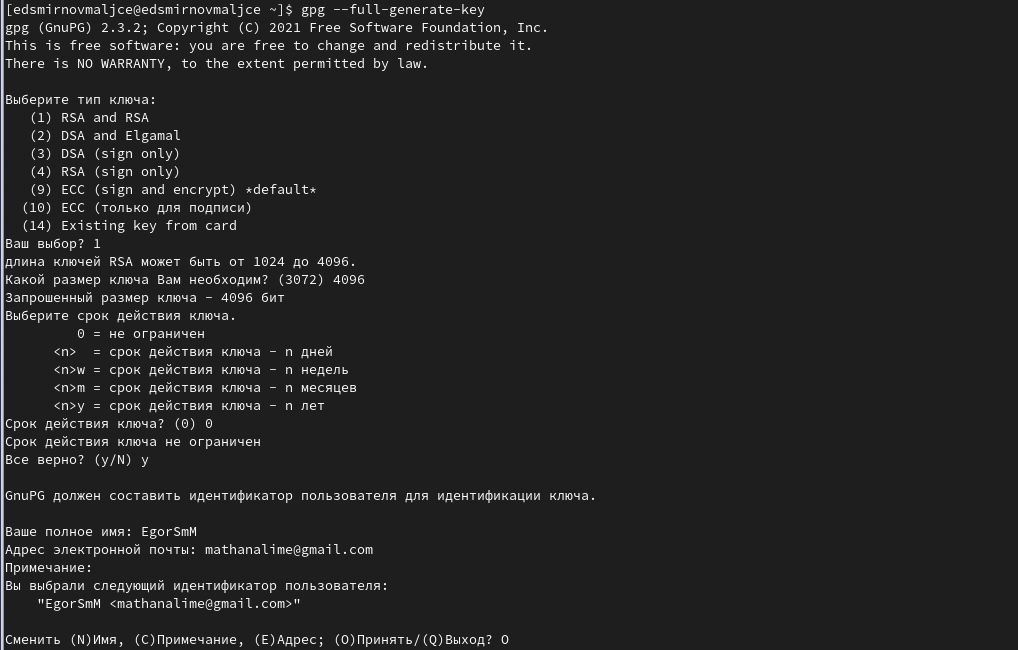


Рисунок Генерация pgp-ключа

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

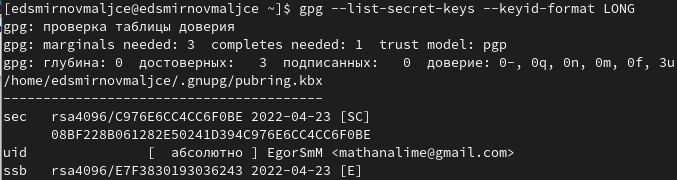


Рисунок Копирование pgp-ключа

1. Скопировал ssh-ключ и ввел его на Github

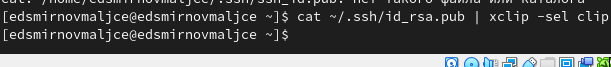


Рисунок Копирование ssh-ключа

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

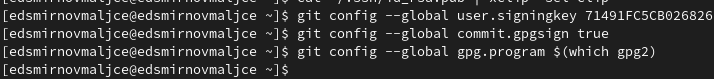


Рисунок Настройка автоматических подписей

1. Авторизировался на Github.



Рисунок Переход к авторизации через командную строку

1. Создал репозиторий на основе шаблона рабочего пространства.

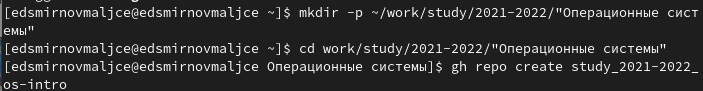


Рисунок Создание директории и переход в нее

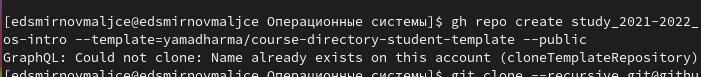


Рисунок Копирование шаблона репозитория

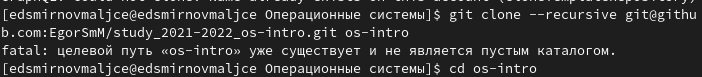


Рисунок Создание пути os-intro

1. Настроил репозиторий.



Рисунок Удаление лишнего файла



Рисунок Создание каталогов

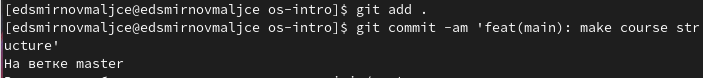


Рисунок Подготовка к отправлению файла на сервер



Рисунок Отправление файла на сервер

**Выводы:**

1. Я научился создавать репозиторий на Github с помощью командной строки.
2. 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.

1. 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.
2. При единоличной работе с VCS берешь нужную версию, вносишь правки и добавляешь коммит.
3. 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.
4. 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.
5. 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

1. 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.

1. Ветви необходимы в случае, если есть несколько путей развития программы, поэтому из одной версии получаются сразу несколько.
2. Игнорировать файлы можно с помощью команды gitignore для того, чтобы не забивать хранилище мусором.