Отчет по лабораторной работе №2

Первоначальная настройка git

Галацан Николай, НПИбд-01-22

Содержание

# 1 Цель работы

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

# 2 Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом.

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.

Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями.

Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.

# 3 Выполнение лабораторной работы

Устанавливаю git: dnf install git

Создаю учетную запись на сайте репозитория Github (https://github.com/) и заполняю основные данные (рис. [1](#fig:11))

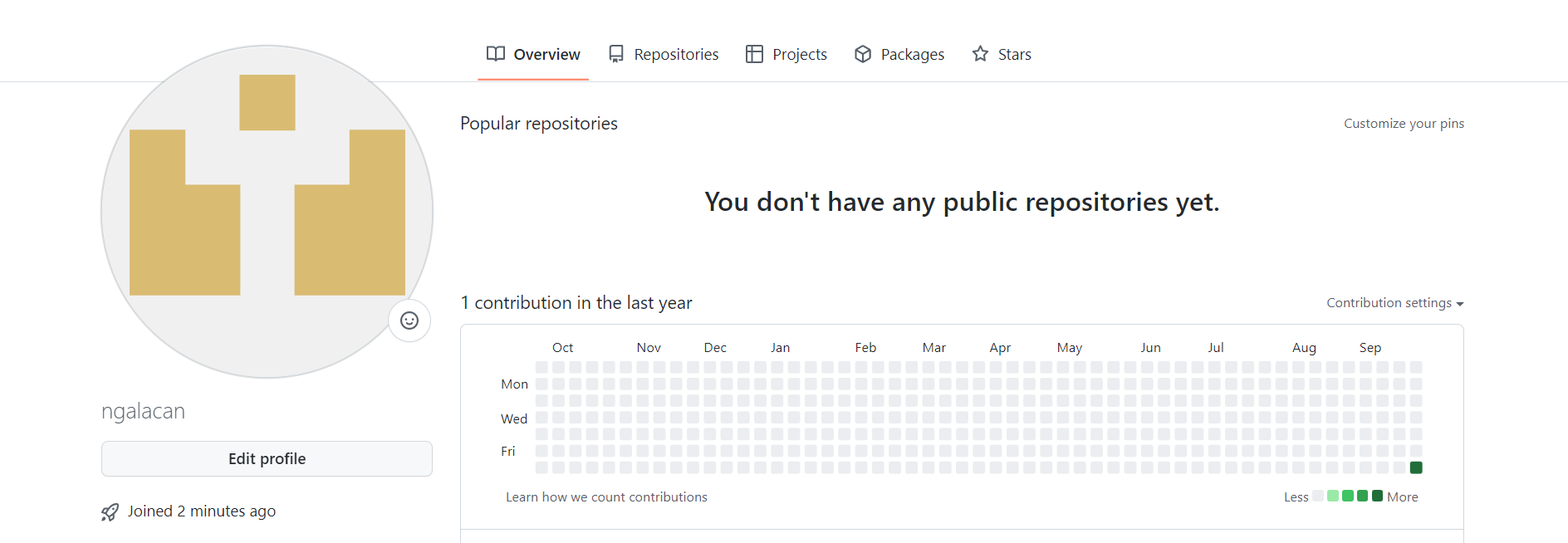


Figure 1: Аккаунт на Github

Делаю предварительную конфигурацию git (рис. [2](#fig:12)). Открываю терминал и ввожу команды, указав имя и email владельца репозитория:

git config --global user.name "ngalacan"

git config --global user.email "1032225763@pfur.ru"

Настраиваю utf-8 в выводе сообщений git:

git config --global core.quotepath false

Задаю имя начальной ветки (master):

git config --global init.defaultBranch master

Параметр autocrlf:

git config --global core.autocrlf input

Параметр safecrlf:

git config --global core.safecrlf warn

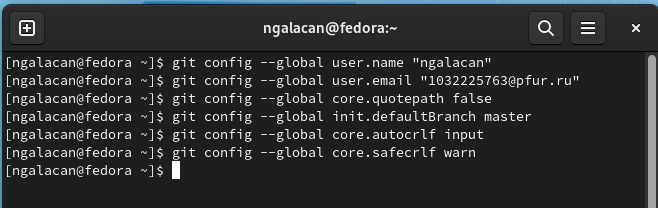


Figure 2: Предварительная конфигурация git

Генерирую ключи SSH для идентификации пользователя на сервере репозиториев (рис. [3](#fig:13)).

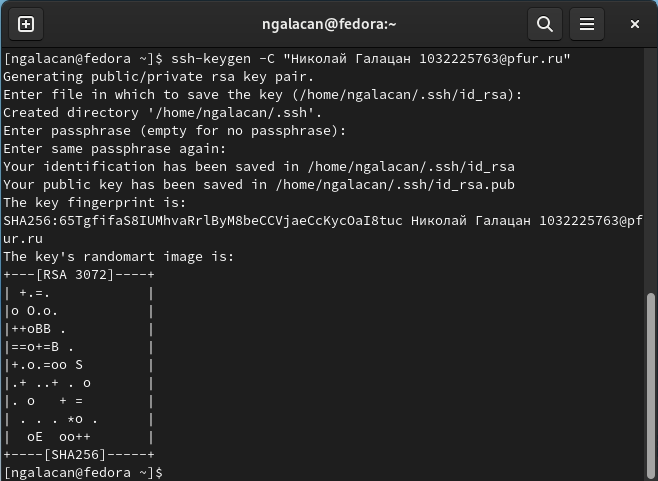


Figure 3: Генерация ключей

Загружаю открытый ключ на Github. На сайте перехожу *Setting => SSH and GPG keys => New SSH key*. Ввожу в терминал cat ~/.ssh/id\_rsa.pub | xclip -sel clip и копирую ключ. Вставляю в появившееся на сайте поле скопированный ключ и указываю имя (рис. [4](#fig:14)).

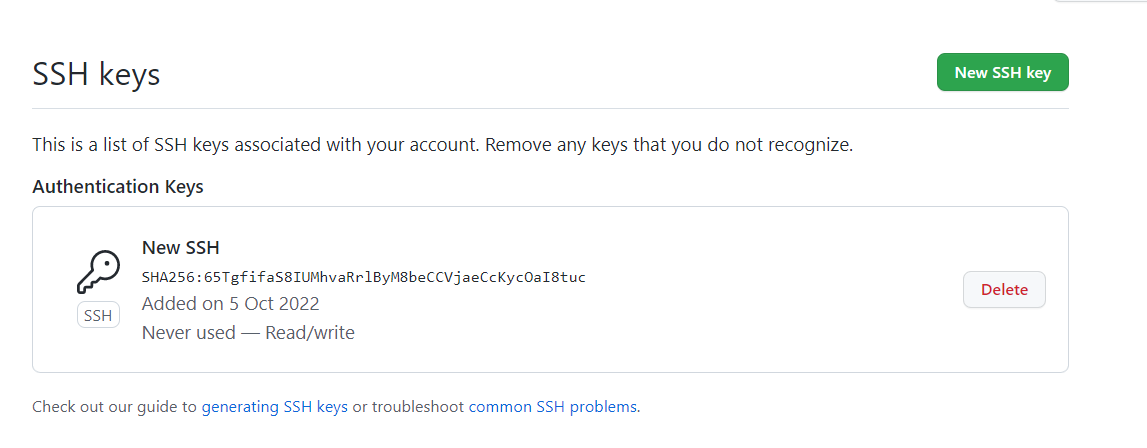


Figure 4: Созданный SSH ключ на сайте

Создаю ключи GPG с помощью команды gpg --full-generate-key и выбираю из предложенных опций тип *RSA and RSA*, размер *4096*; срок действия - значение по умолчанию — 0. Заполняю личную информацию (рис. [5](#fig:1)).

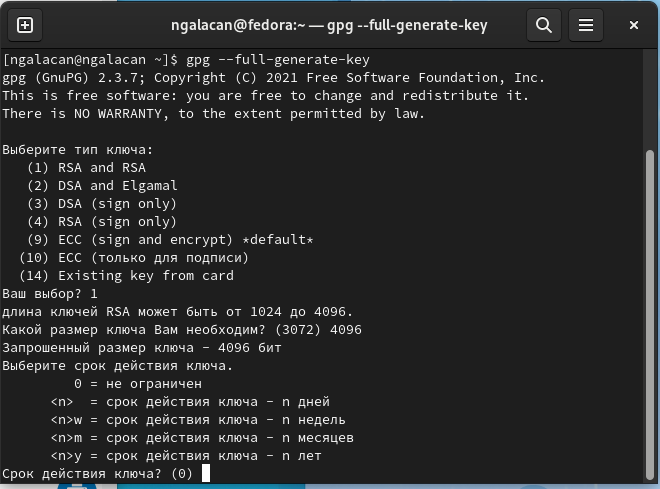


Figure 5: Создание ключа GPG

Добавляю ключ GPG на github, предварительно выведя список ключей и скопировав отпечаток приватного ключа (рис. [6](#fig:2), [7](#fig:3)).

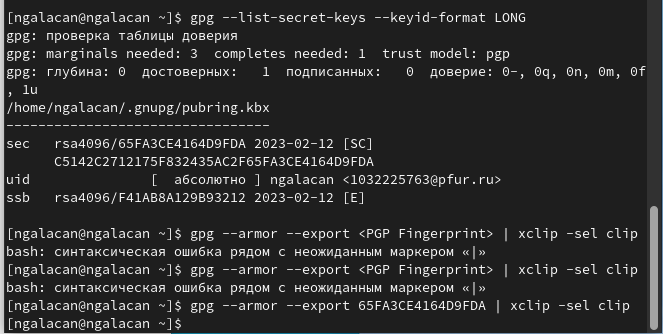


Figure 6: Копирование ключа GPG

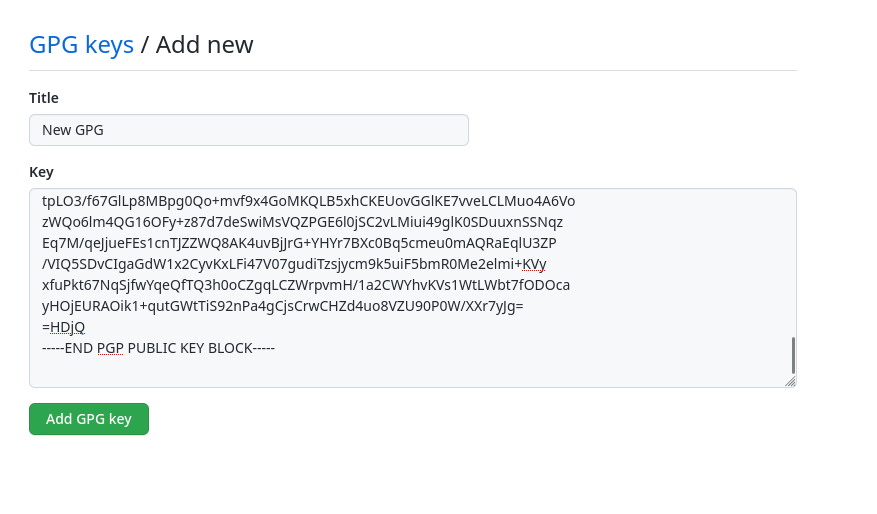


Figure 7: Созданный ключ GPG на сайте

Настраиваю автоматические подписи коммитов git (рис. [8](#fig:4)).

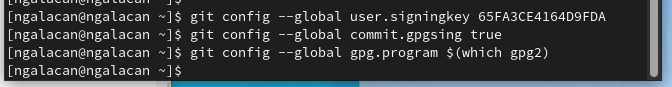


Figure 8: Настройка подписей коммитов git

Устанавливаю и настраиваю gh для авторизации в github, привязываю устройство (рис. [9](#fig:6)).

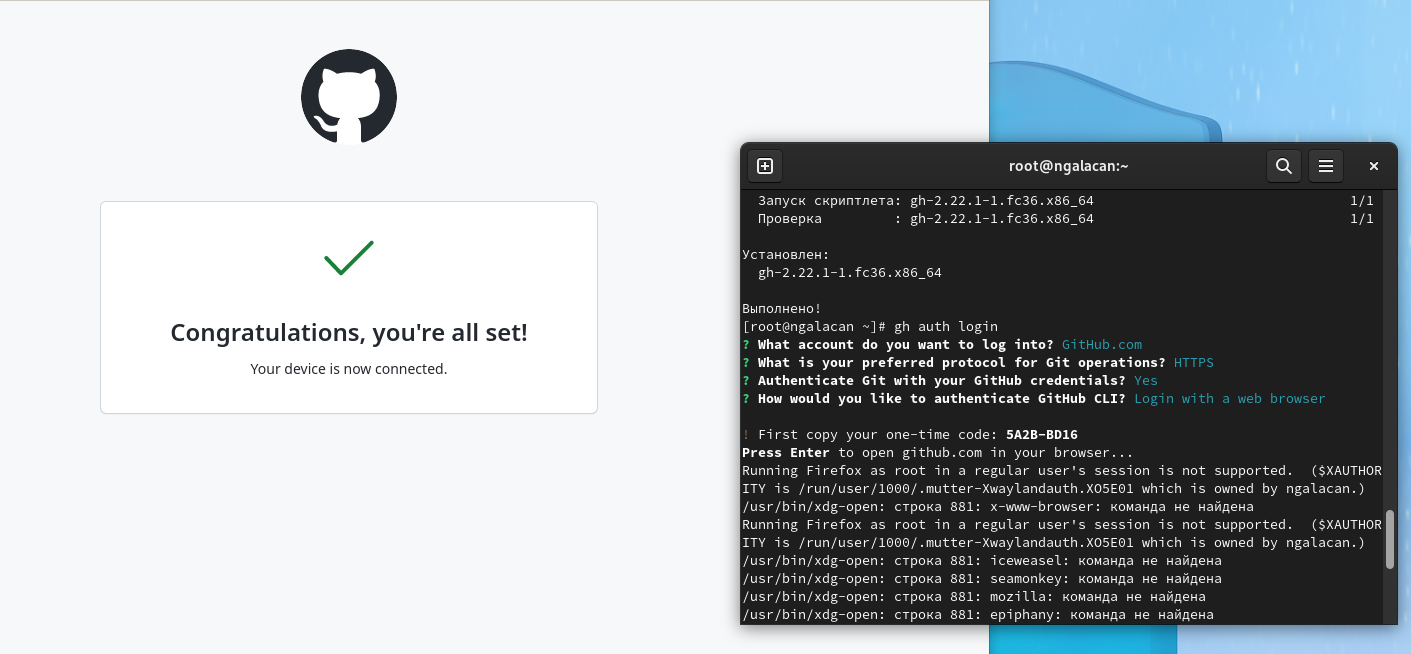


Figure 9: Настройка gh

Создаю через терминал каталог для предмета “Операционные системы” и клонирую репозиторий с помощью команд

mkdir -p ~/work/study/2022-2023/"Операционные системы"  
cd ~/work/study/2022-2023/"Операционные системы"  
gh repo create study\_2022-2023\_os-intro --template=yamadharma/course-directory-student-template --public  
git clone --recursive git@github.com:<owner>/study\_2022-2023\_os-intro.git os-intro

(рис. [10](#fig:7)).

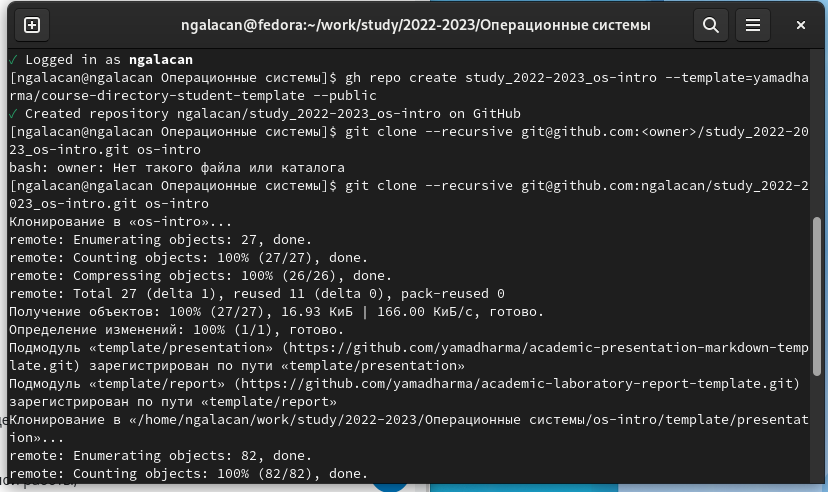


Figure 10: Создание каталога для предмета и клонирование репозитория

Перехожу в каталог курса и удаляю лишние файлы, используя команды cd и rm. Создаю необходимые каталоги и отправляю файлы на сервер (рис. [11](#fig:8)).

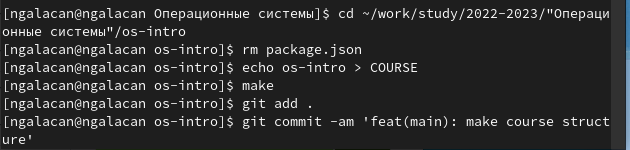


Figure 11: Настройка каталога курса

Проверяю, отправились ли файлы на github (рис. [12](#fig:9)).

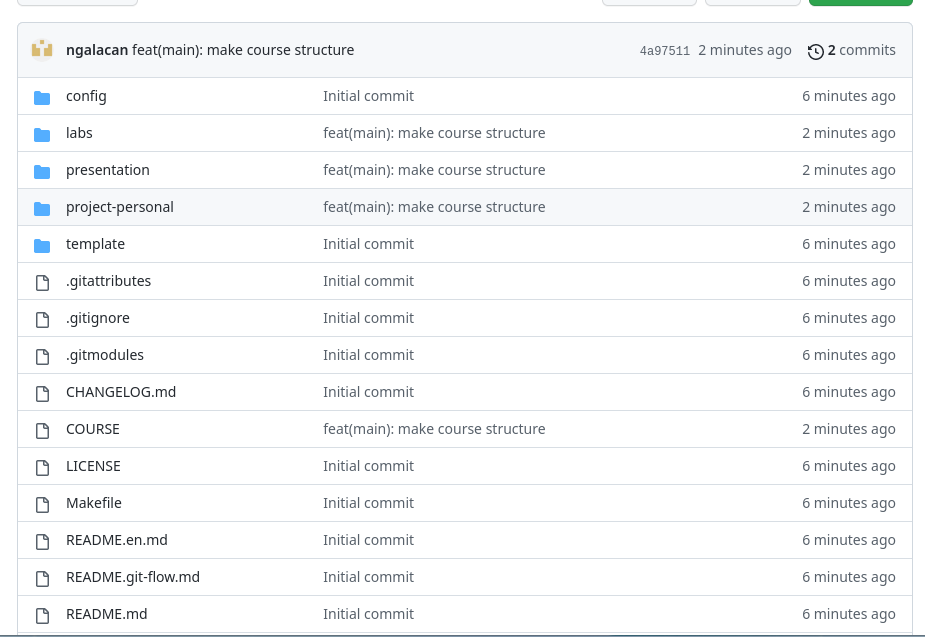


Figure 12: Проверка репозитория на странице github

Файлы соответствуют.

# 4 Выводы

Была изучена идеология и применение средств контроля версий. Была настроена система git. Приобретены практические навыки по работе с системой git.

# 5 Ответы на контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

Системы контроля версий (Version Control System, VCS) - система регистрации изменений в одном или нескольких файлах. Применяются при работе нескольких человек над одним проектом.

1. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия.

Хранилище - область на локальном или удаленном репозитории, где хранятся файлы проекта.

Сommit - внесение правок или изменение файлов в репозитории.

История - история изменений в репозитории.

Рабочая копия - копия главного удаленного репозитория, в которой работает пользователь.

1. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида.

Централизованная модель предполагает наличие единого репозитория для хранения файлов. В отличие от централизованных, в распределённых (децентрализованных) системах контроля версий центральный репозиторий не является обязательным. Примеры централизованных VCS: CVS, Subversion; примеры децентрализованных: Git, Bazaar.

1. Опишите действия с VCS при единоличной работе с хранилищем.

Сохранение файлов на удаленном репозитории, коммит, отправка на сервер и др.

1. Опишите порядок работы с общим хранилищем VCS.

Обновление до текущей версии репозитория (например, git pull), сохранение файлов на удаленном репозитории, коммит, отправка на сервер и др.

1. Каковы основные задачи, решаемые инструментальным средством git?

Объединение изменений, выбор нужной версии, отменение изменений или блокировка и др.

1. Назовите и дайте краткую характеристику командам git.

Создание основного дерева репозитория: git init

Получение обновлений 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

добавить все изменённые и/или созданные файлы и/или каталоги: git add .

сохранить все добавленные изменения и все изменённые файлы: git commit -am ‘Описание коммита’

1. Приведите примеры использования при работе с локальным и удалённым репозиториями.

git pull - для обновления локального репозитория.

git status - для просмотра изменений локального репозитория.

git add . - для добавления всех изменений в удаленный репозиторий.

git push - для отправки всех изменений на удаленный репозиторий.

1. Что такое и зачем могут быть нужны ветви (branches)?

Ветви - последовательность коммитов и изменений. Они нужны для работы нескольких пользователей над одним проектом.

1. Как и зачем можно игнорировать некоторые файлы при commit?

С помощью команды git rm имена\_файлов. Это нужно для того, чтобы файл не отправился в удаленный репозиторий (например, если работа не завершена или он не нужен, то и в коммит он попасть не должен).