Отчёт по лабораторной работе № 2

НБИбд-01-23

Анастасия Романовна Зинченко

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

1. Установка программного обеспечения а. Установка git б. Установка gh
2. Базовая настройка git
3. Создайте ключи ssh
4. Создайте ключи pgp
5. Настройка github
6. Добавление PGP ключа в GitHub
7. Настройка автоматических подписей коммитов git
8. Настройка gh
9. Шаблон для рабочего пространства а. Сознание репозитория курса на основе шаблона б. Настройка каталога курса
10. Контрольные вопросы

3 Выполнение лабораторной работы

Установила git с помощью команды dnf install git (рис. 3.1).

Рис. 3.1: Установка git

Установила gh с помощью команды dnf install gh (рис. 3.2).

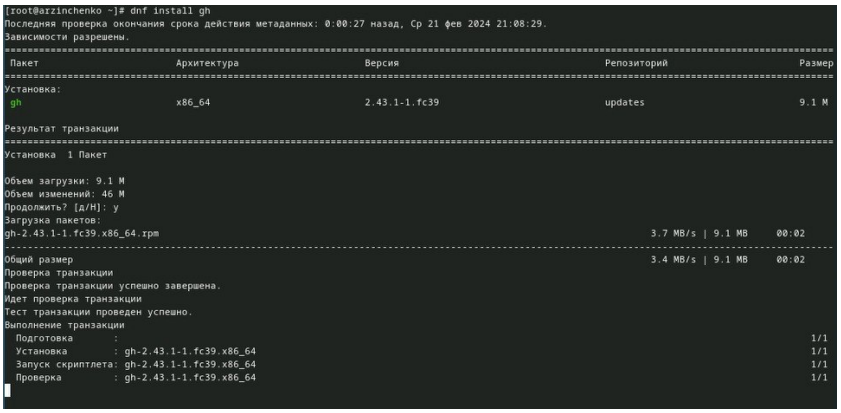

Рис. 3.2: Установка gh

Задала имя и email своего репозитория с помощью команд git config –global user.name “Anastasiia” git config –global user.email “zinchenkoa06zinchenko@yandex.ru” (рис. 3.3).

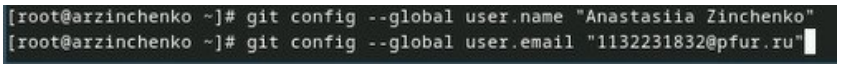

Рис. 3.3: Установка gh

Настроила utf-8 в выводе сообщений git с помощью команды git config –global core.quotepath false (рис. 3.4).

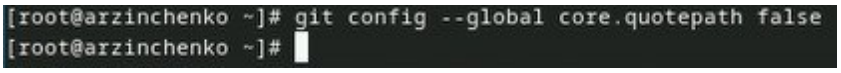

Рис. 3.4: Настройка utf-8

Задала имя начальной ветки с помощью команды git config –global init.defaultBranch master (рис. 3.5).

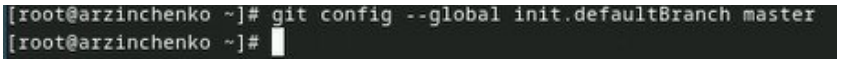

Рис. 3.5: Ветка master

Задала параметр autocrlf с помощью команды git config –global core.autocrlf input (рис. 3.6).

Рис. 3.6: Параметр autocrlf

Задала параметр safecrlf с помощью команды git config –global core.safecrlf warn (рис. 3.7).

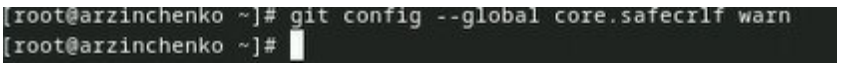

Рис. 3.7: Параметр safecrlf

Создала ключ ssh по алгоритму rsa с ключём размером 4096 бит с помощью команды ssh-keygen -t rsa -b 4096 (рис. 3.8).

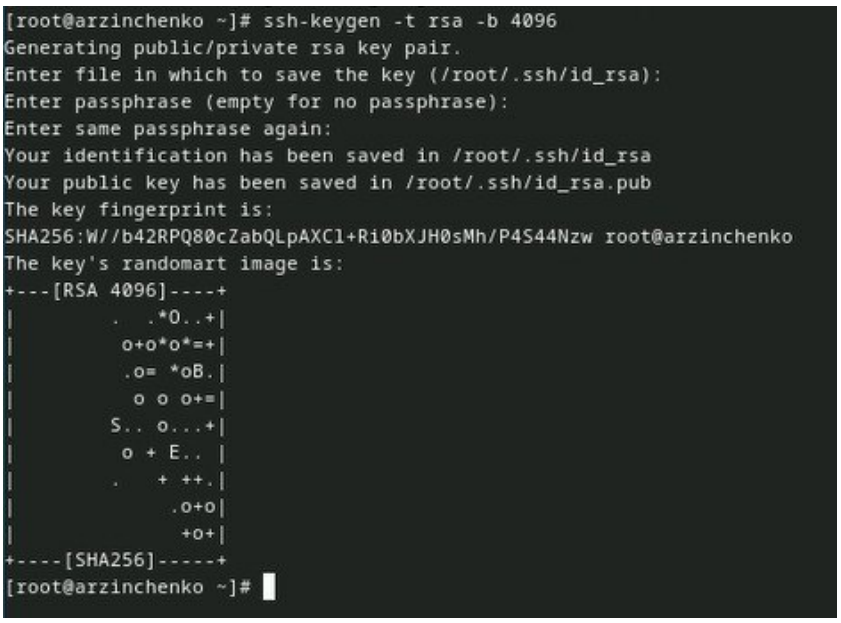

Рис. 3.8: Ключ ssh по алгоритму rsa

Создала ключ ssh по алгоритму ed25519 с помощью команды ssh-keygen -t ed25519 (рис. 3.9).

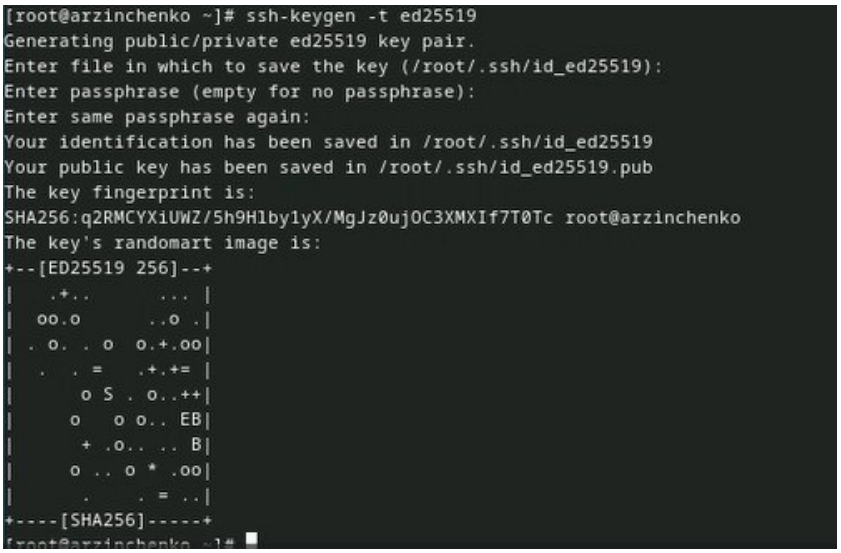

Рис. 3.9: Ключ ssh по алгоритму ed25519

Сгенерировала ключ gpg с помощью команды gpg –full-generate-key (рис. 3.10).

Рис. 3.10: Генерация ключа

Из предложенных опций выбирала тип RSA and RSA; размер 4096; срок действия; имя; адрес электронной почты

У меня уже была создана учётная запись на Github и были заполнены основные данные (рис. 3.11).

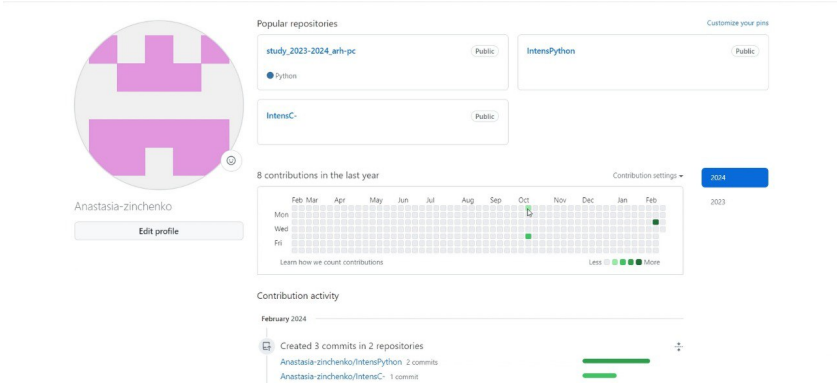

Рис. 3.11: Учётная запись Github

Вывела список ключей и скопировала отпечаток приватного ключа с помощью команды gpg –list-secret-keys –keyid-format LONG (рис. 3.12).

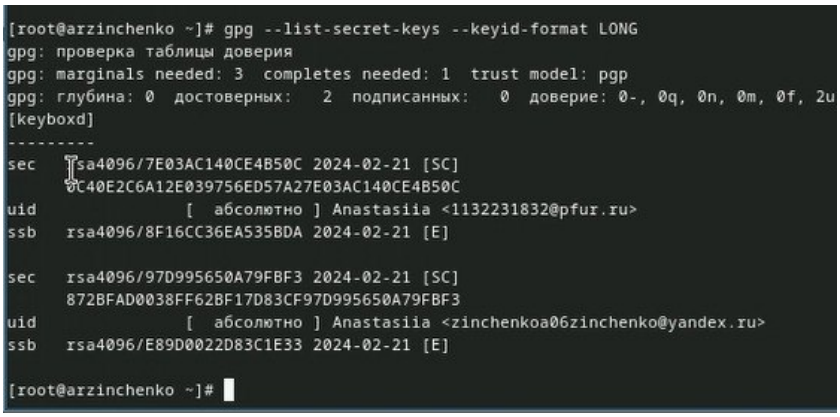

Рис. 3.12: Список ключей

Скопировала сгенерированный PGP ключ в буфер обмена с помощью команды gpg –armor –export | xclip -sel clip. Перешла в настройки GitHub, нажала на кнопку New GPG key и вставила полученный ключ в поле ввода (рис. 3.13).

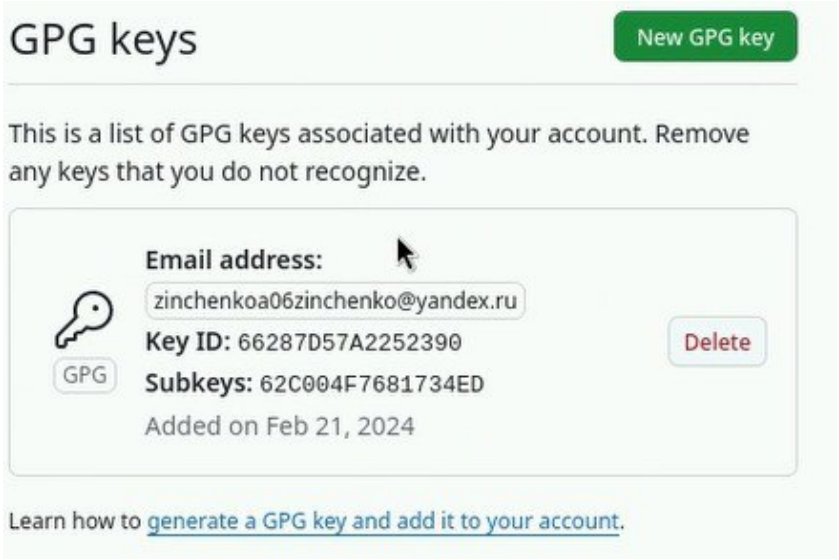

Рис. 3.13: Список ключей

Используя zinchenkoa06zinchenko@yandex.ru, указала Git применять его при подписи коммитов с помощью команд git config –global user.signingkey git config –global commit.gpgsign true git config –global gpg.program $(which gpg2) (рис. 3.14).

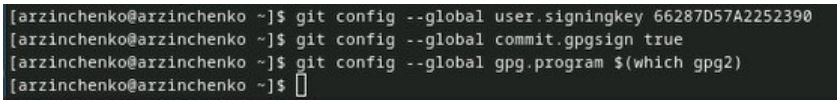

Рис. 3.14: Подписи коммиттов

Я авторизовалась с помощью команды gh auth login (рис. 3.15).

Рис. 3.15: Авторизация

Я создала 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:/study_2022-2023_os-intro.git os-intro (рис. 3.16).

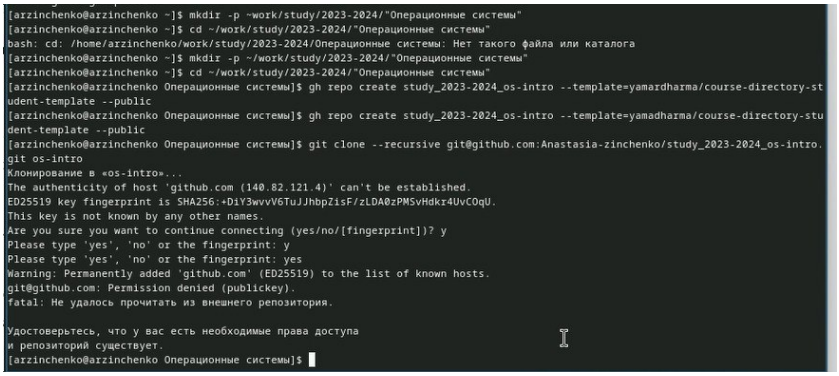

Рис. 3.16: Создание репозитория

Для того чтобы насторить каталог курса я перешла в каталог курса с помощью команды cd ~/work/study/2022-2023/“Операционные системы”/os-intro (рис. 3.17).

Рис. 3.17: Создание репозитория

Удалила лишние файлы с помощью команды rm package.json (рис. 3.18).

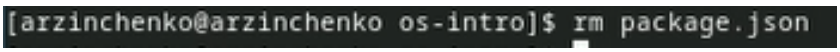

Рис. 3.18: Удаление лишних файлов

Создала необходимые каталоги с помощью команд echo os-intro > COURSE make (рис. 3.19).

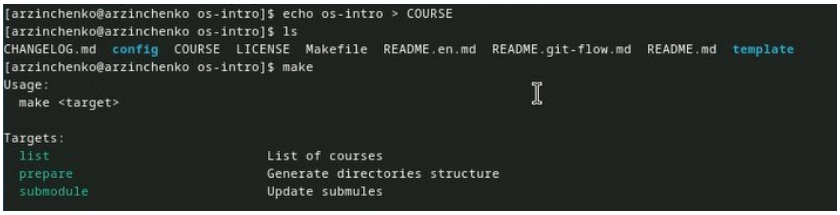

Рис. 3.19: Создание каталогов

Отправила файлы на сервер с помощью команд git add . git commit -am ‘feat(main): make course structure’ git push (рис. 3.20).

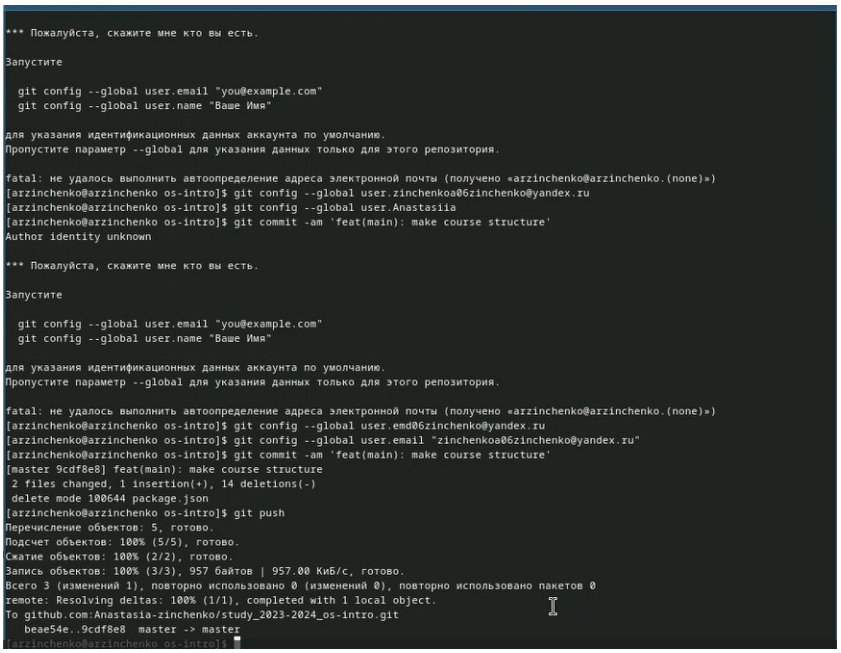

Рис. 3.20: Создание каталогов

Проверка файлов на сервере (рис. 3.21).

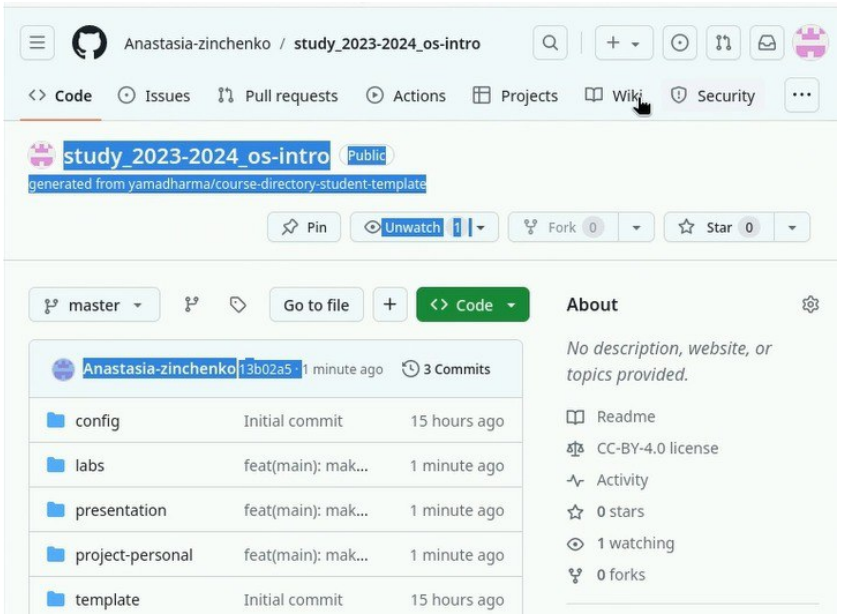

Рис. 3.21: Проверка

Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Это програмное обеспечение для облегчения работы с информацией, которая изменяется. Они позволяют хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, а также определять кто и когда сделал какие - либо изменения.
2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище -место хранения всех версий и служебной информации. Commit - процесс создания новой версии. История -место, где сохранаются все коммиты, по которым можно посмотреть данные о коммитах. Рабочая копия - текущее состояние файлов проекта, основанное на версии, загруженной из хранилища.
3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида. Централизованные VCS: одно основное хранилище всего проекта и каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет, а затем добавляет свои изменения обратно. Децентрализованные VCS: у каждого пользователя свой вариант репозитория.
4. Опишите действия с VCS при единоличной работе с хранилищем.
5. Опишите порядок работы с общим хранилищем VCS.
6. Каковы основные задачи, решаемые инструментальным средством git? 1. Хранение информации о всех изменениях в коде. 2. Обеспечение удобства работы над проектом в команде.
7. Назовите и дайте краткую характеристику командам git. 1. git-version (проверка версии Git). 2. git init - инициализация текущего рабочего каталога как Git репозиторий. 3. git clone - копирование существующего удаленного Git - репозитория. 4. git remote - просмотр списка текущих удаленных репозиториев. 5. git commit -am “Commit message” - сжимание всех индексированных

файлов и отправка коммитов. 6. git branch - просмотр списка текущих веток.

8. Приведите примеры использования при работе с локальным и удалённым репозиториями.
9. Что такое и зачем могут быть нужны ветви (branches)? Ветви нужны для того, чтобы программисты могли одновременно работать над одним и тем же файлом, не мешая друг другу.
10. Как и зачем можно игнорировать некоторые файлы при commit? Игнорируемы файлы - артефакты сборки и файлы, генерируемые машиной из исходных файлв в репозитории, либо файлы, которые не должны попадать в коммиты.

4 Выводы

Я изучила идеологию и применение средств контроля версий и освоила умения по работе с git.

Список литературы