Отчёт по лабораторной работе №2

Первоначальная настройка git

Чекмарев Александр Дмитриевич | Группа НПИбд-02-23

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

## 2.1 Установка программного обеспечения

Установка git • Установим git: dnf install git

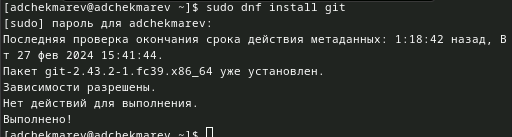


Рис 2.1.1: установка git

Установка gh • Установим gh: dnf install gh

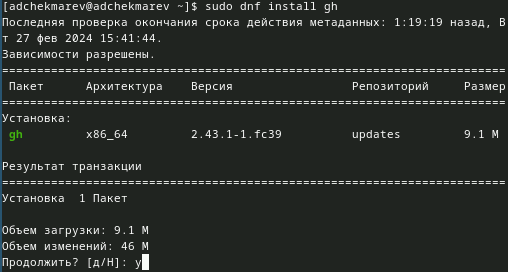


Рис 2.1.2: установка gh

## 2.2 Базовая настройка git

Зададим имя и email владельца репозитория: *git config –global user.name “Name Surname”*

Рис 2.2.1: запись имени

Рис 2.2.1: запись имени

*git config –global user.email “work@mail”*

Рис 2.2.2: запись почты

Рис 2.2.2: запись почты

Настроим utf-8 в выводе сообщений git: *git config –global core.quotepath false*

Рис 2.2.3: настройка utf-8

Рис 2.2.3: настройка utf-8

Настроим верификацию и подписание коммитов git (см. Верификация коммитов git с помощью GPG). Зададим имя начальной ветки (будем называть её master): *git config –global init.defaultBranch master*

Рис 2.2.4: задача начальной ветки

Рис 2.2.4: задача начальной ветки

Параметр autocrlf: *git config –global core.autocrlf input*

Рис 2.2.5: настройка параметра

Рис 2.2.5: настройка параметра

Параметр safecrlf: *git config –global core.safecrlf warn*

Рис 2.2.6: настройка параметра

Рис 2.2.6: настройка параметра

## 2.3 Создание ключа ssh

по алгоритму rsa с ключём размером 4096 бит: *ssh-keygen -t rsa -b 4096*

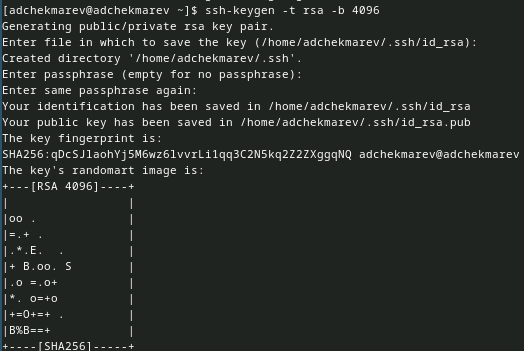


Рис 2.3.1: создание ключа с конкретным размером

по алгоритму ed25519: *ssh-keygen -t ed25519*

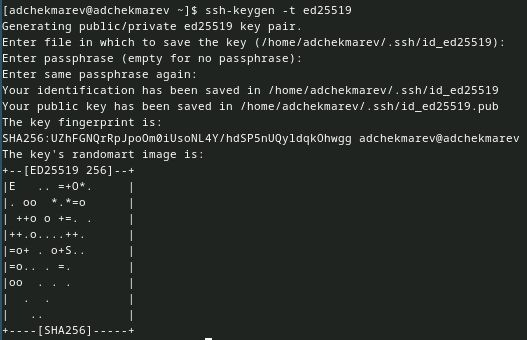


Рис 2.3.2: создание ключа ed25519

## 2.4 Создание ключа pgp

• Генерируем ключ: gpg --full-generate-key  
• Из предложенных опций выбираем:  
 ◦ тип RSA and RSA;  
 ◦ размер 4096;  
 ◦ выберите срок действия; значение по умолчанию — 0 (срок действия не истекает никогда).  
• GPG запросит личную информацию, которая сохранится в ключе:  
 ◦ Имя (не менее 5 символов).  
 ◦ Адрес электронной почты.  
 ▪ При вводе email убедитесь, что он соответствует адресу, используемому на GitHub.  
 ◦ Комментарий. Можно ввести что угодно или нажать клавишу ввода, чтобы оставить это поле пустым

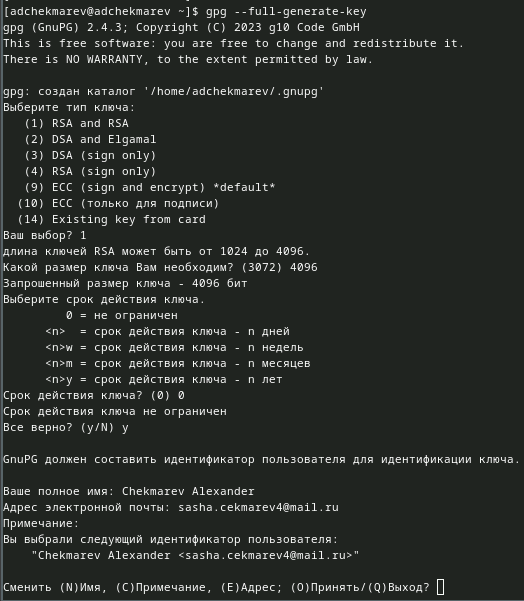


Рис 2.4.1: процесс перед генерацией ключа

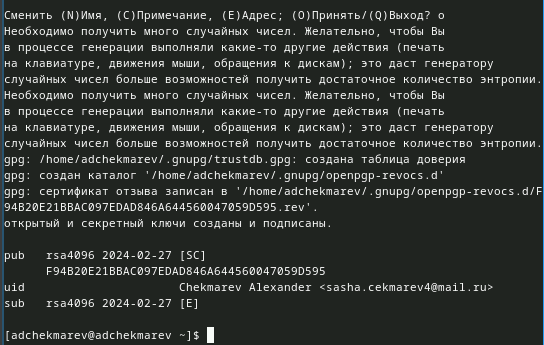


Рис 2.4.2: сгенерированный ключ

## 2.5 Настройка github

Чтобы работать с github, нам нужно будет создать аккаунт на сайте https://github.com/. У меня аккаунт уже есть с прошлого семестра.



Рис 2.5.1: демонстрация профиля на github

## 2.6 Добавление PGP ключа в GitHub

• Выводим список ключей и копируем отпечаток приватного ключа: \*gpg --list-secret-keys --keyid-format LONG\*

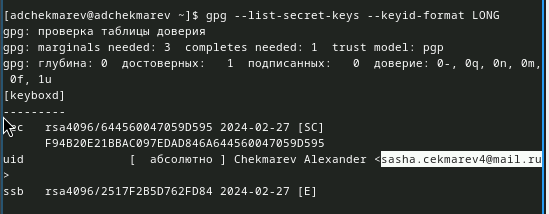


Рис 2.6.1: вывод списка ключей

• Отпечаток ключа — это последовательность байтов, используемая для идентификации более длинного, по сравнению с самим отпечатком ключа.  
• Формат строки: sec Алгоритм/Отпечаток\_ключа Дата\_создания [Флаги] [Годен\_до] ID\_ключа  
• Cкопируем ваш сгенерированный PGP ключ в буфер обмена: \*gpg --armor --export <PGP Fingerprint> | xclip -sel clip\*

Рис 2.6.2: сгенерированный PGP ключ

Рис 2.6.2: сгенерированный PGP ключ

Перейдем в настройки GitHub (https://github.com/settings/keys), нажмем на кнопку New GPG key и вставим полученный ключ в поле ввода.

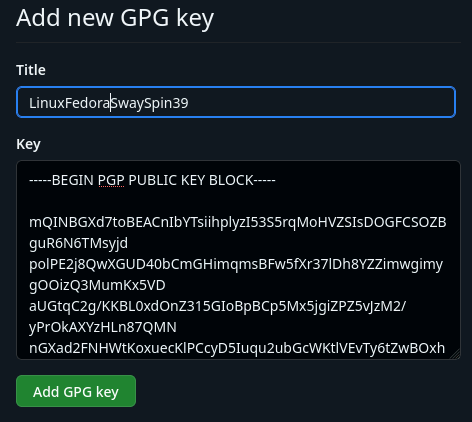


Рис 2.6.3: создание нового GPG ключа на гитхабе

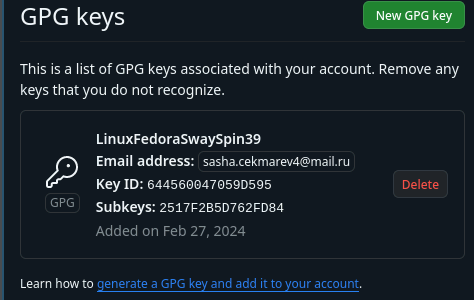


Рис 2.6.4: демонстрация завершения работы с GPG ключом

## 2.7 Настройка автоматических подписей коммитов git

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

*git config –global user.signingkey*  *git config –global commit.gpgsign true* *git config –global gpg.program $(which gpg2)*

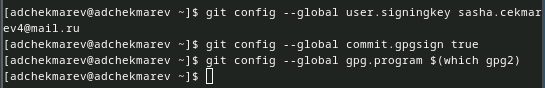


Рис 2.7.1: настройка подписей коммитов git

## 2.8 Настройка gh

Для начала необходимо авторизоваться: gh auth login Утилита задаст несколько наводящих вопросов. Авторизоваться можно через браузер.

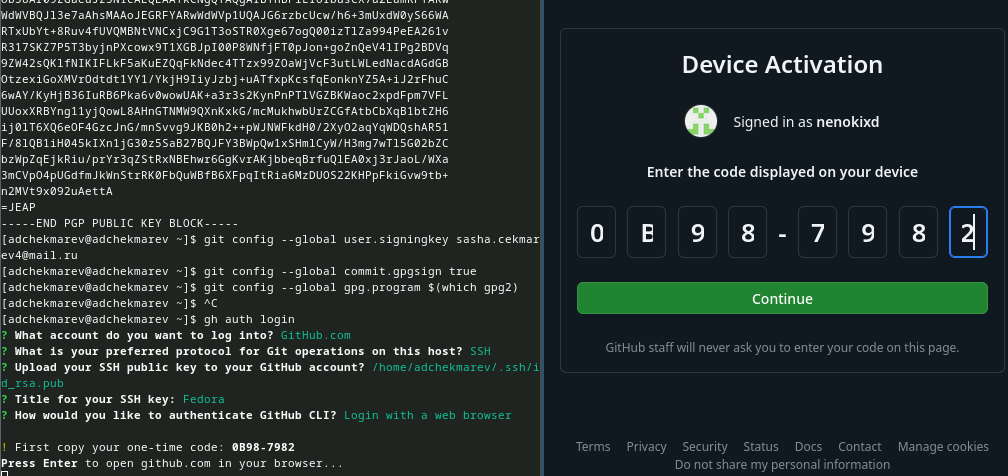


Рис 2.8.1: демонстрация настройки gh

## 2.9 Шаблон для рабочего пространства

Рабочее пространство для лабораторной работы Репозиторий: https://github.com/yamadharma/course-directory-student-template.

Сознание репозитория курса на основе шаблона Необходимо создать шаблон рабочего пространства (см. Рабочее пространство для лабораторной работы). Например, для 2023–2024 учебного года и предмета «Операционные системы» (код предмета os-intro) создание репозитория примет следующий вид: *mkdir -p ~/work/study/2023-2024/“Операционные системы”*

Рис 2.9.1: создание каталога

Рис 2.9.1: создание каталога

*cd ~/work/study/2023-2024/“Операционные системы”*

Рис 2.9.2: переход в каталог

Рис 2.9.2: переход в каталог

*gh repo create study\_2023-2024\_os-intro –template=yamadharma/course-directory-student-template –public*

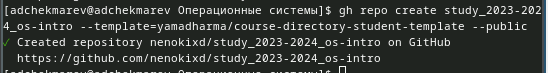


Рис 2.9.3: создание репозитория

*git clone –recursive git@github.com:/study\_2023-2024\_os-intro.git os-intro*

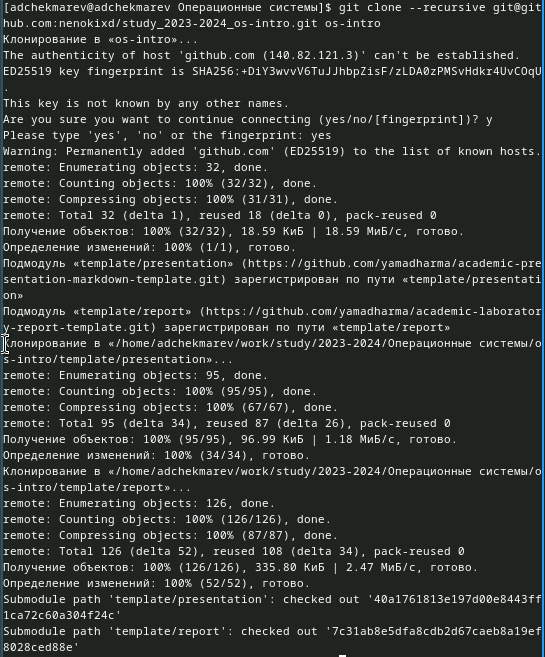


Рис 2.9.4: клонирование репозитория

**Настройка каталога курса**

Перейдите в каталог курса: *cd ~/work/study/2023-2024/“Операционные системы”/os-intro*

Рис 2.9.5: переход в другой каталог

Рис 2.9.5: переход в другой каталог

Удалите лишние файлы: *rm package.json*

Рис 2.9.6: удаление файла .json

Рис 2.9.6: удаление файла .json

Создайте необходимые каталоги: *echo os-intro > COURSE* *make*

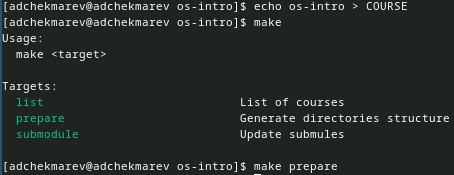


Рис 2.9.7: создание каталогов

Отправьте файлы на сервер:

*git add .*

*git commit -am ‘feat(main): make course structure’*

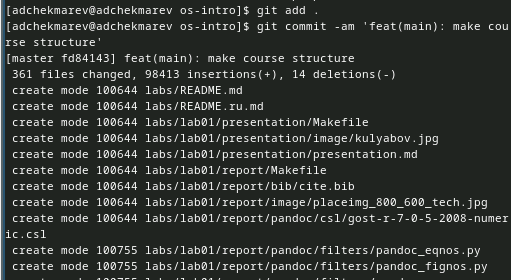


Рис 2.9.8: сохранение и выбор файлов для отправки на сервер

*git push*

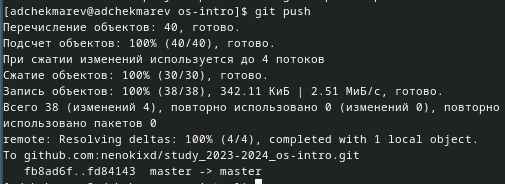


Рис 2.9.9: отправка файлов на сервер

# 3 Контрольные вопросы

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

* Системы контроля версий (VCS) - это инструменты, которые отслеживают изменения в файловой системе с течением времени. Они предназначены для управления изменениями в коде и других файлах проекта, позволяя разработчикам работать над проектом одновременно, откатывать изменения, если что-то идет не так, и отслеживать историю изменений.

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

* Хранилище (repository): Это место, где хранятся все файлы и история изменений проекта. Commit: Это операция сохранения изменений в репозитории. При коммите фиксируются все изменения, сделанные с момента предыдущего коммита. История (history): Это записи о всех коммитах, сделанных в репозитории. Рабочая копия (working copy): Это копия файлов из репозитория, с которой вы работаете на вашем компьютере.

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

* Централизованные VCS имеют одно основное хранилище, к которому подключаются все клиенты. Децентрализованные VCS позволяют каждому клиенту иметь собственное полноценное хранилище. Примеры централизованных VCS: Subversion (SVN). Примеры децентрализованных VCS: Git, Mercurial.

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

При единоличной работе с хранилищем в VCS вы создаете, изменяете и фиксируете изменения в рабочей копии и затем коммитите их в репозиторий.

1. Опишите порядок работы с общим хранилищем VCS.

Порядок работы с общим хранилищем VCS включает получение последних изменений из репозитория (pull), внесение своих изменений, фиксацию изменений (commit) и отправку их в репозиторий (push).

1. Каковы основные задачи, решаемые инструментальным средством git?

Основные задачи инструмента git:

Отслеживание изменений в файлах.  
Управление версиями проекта.  
Работа с удаленными репозиториями.  
Ветвление и слияние изменений.  
Работа с ветками.

1. Назовите и дайте краткую характеристику командам git.

Некоторые команды git:

git init: Создает новый репозиторий.  
git add: Добавляет файлы в индекс для последующего коммита.  
git commit: Фиксирует изменения в репозитории.  
git push: Отправляет изменения в удаленный репозиторий.  
git pull: Получает изменения из удаленного репозитория и объединяет их с текущей веткой.

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

Примеры использования при работе с локальным и удаленным репозиториями:

Локальный: Создание нового репозитория с помощью git init.  
Удаленный: Клонирование существующего удаленного репозитория с помощью git clone.

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

Ветви (branches) - это параллельные линии разработки в репозитории, которые позволяют работать над разными фичами или исправлениями, не затрагивая основную ветку. Они могут быть нужны, чтобы изолировать разные функциональные изменения или исправления ошибок.

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

Файлы могут быть проигнорированы при коммите с помощью файла .gitignore. В этом файле перечисляются шаблоны файлов или папок, которые не должны быть добавлены в репозиторий.

# 4 Выводы

Я Изучил идеологию и применение средств контроля версий, а также освоил умения по работе с git.

# Список литературы