Лабораторная работа № 2.

Операционные системы

Шулуужук Айраана Вячеславовна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Настроить подписи git.
5. Зарегистрироваться на Github.
6. Создать локальный каталог для выполнения заданий по предмету.

3 Выполнение лабораторной работы

3.1 Установка программного обеспечения

Установка git (рис. 3.1):

```
dnf install git
```

Установка gh:

```
dnf install gh
```

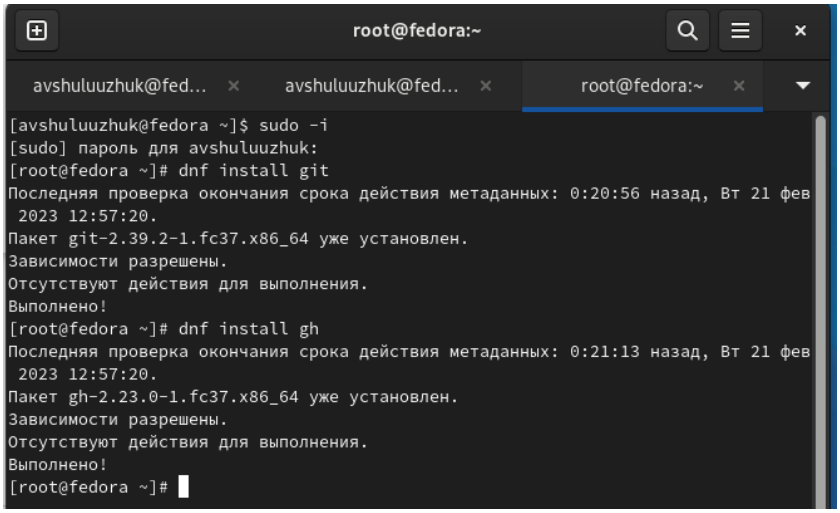

Рис. 3.1: Установка программного обеспечения

3.2 Базовая настройка git

Проделываем предварительную конфигурацию гит (рис. 3.2)

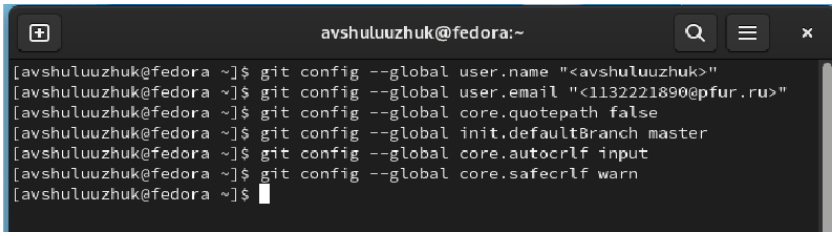

Рис. 3.2: Базовая настройка git

3.3 Создание ключей ssh

Создаем ключи (рис. 3.3)

по алгоритму rsa с ключём размером 4096 бит:

```
ssh-keygen -t rsa -b 4096
```

по алгоритму ed25519:

```
ssh-keygen -t ed25519
```

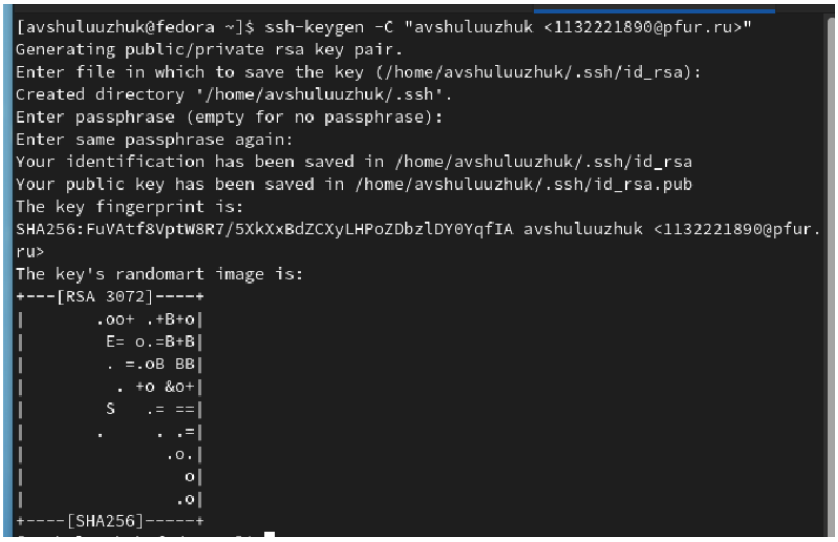

Рис. 3.3: генерация ключей

Загрузим ключи в гит(рис. 3.4), скопируем их, используя команду :

```
xclip -i < ~/.ssh/id_ed25519.pub
```

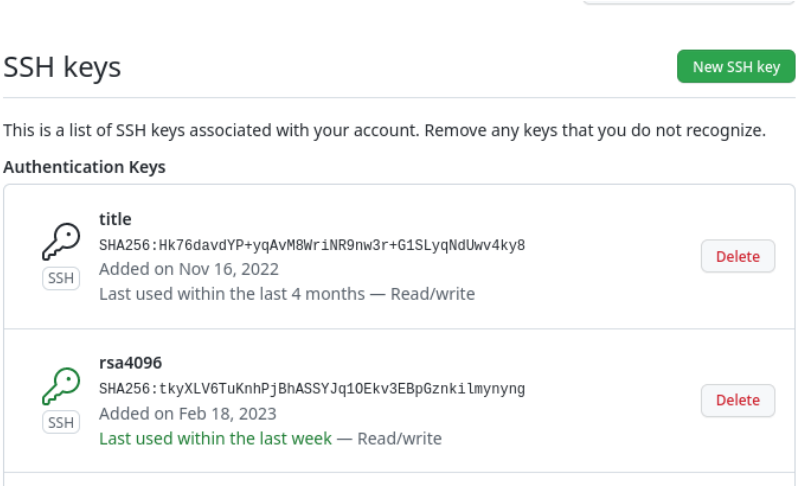

Рис. 3.4: созданный ключ в гит

3.4 Создание pgp ключа

Генерируем ключ (рис. 3.5)

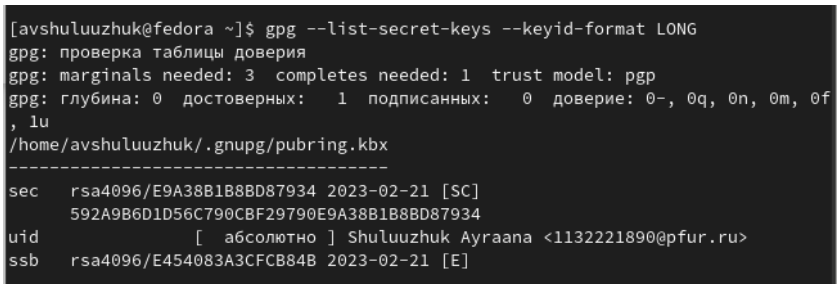

Рис. 3.5: ключ pgp

Скопируем этот ключ и добавляем в гит (рис. 3.6)

```
gpg --armor --export <PGP Fingerprint> | xclip -sel clip
```

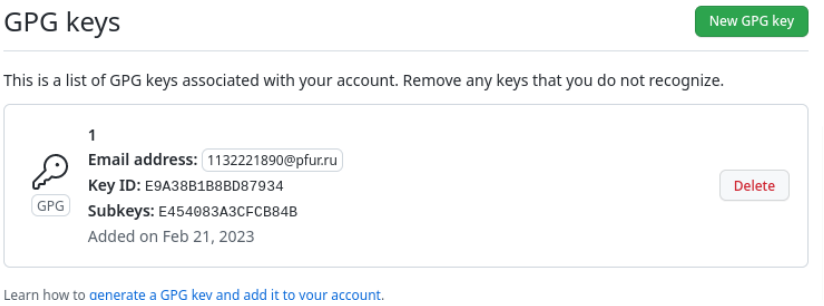

Рис. 3.6: добавление ключа в гит

3.5 Создание репозитория курса на основе шаблона

Используя репозиторий с шаблоном курса, создаем репозиторий “Операционные системы” Клонируем созданный репозиторий (рис. 3.7), ссылку для клонирования копируем на странице созданного репозитория:

```
git clone --recursive
```

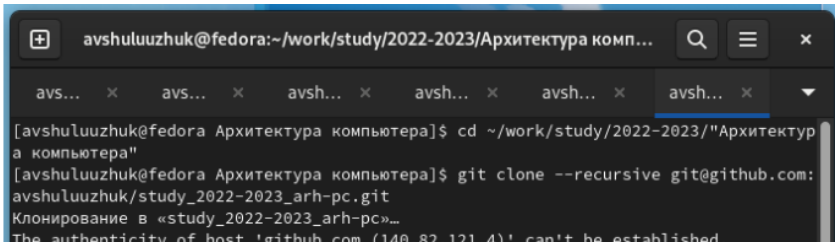

Рис. 3.7: клонирование репозитория

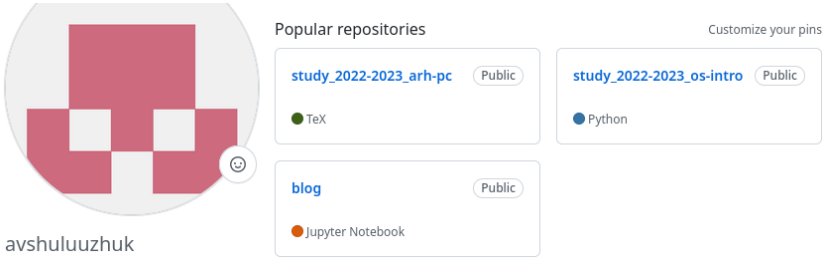

Рис. 3.8: созданный репозитрий

3.6 Настройка каталога курса

Переходим в каталог курса и удаляем лишние файлы (рис. 3.9)

```
rm package.json
```

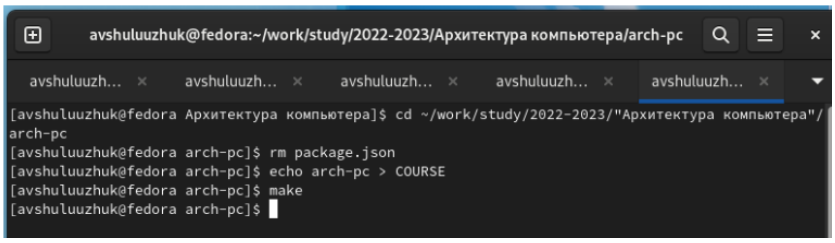

Рис. 3.9: Насторйка каталога курса

Создаем необходимые файлы:

```
echo os-intro > COURSE
make
```

Отправляем файлы на сервер (рис. 3.10)

```
git add .
git commit -am 'feat(main): make course structure'
git push
```

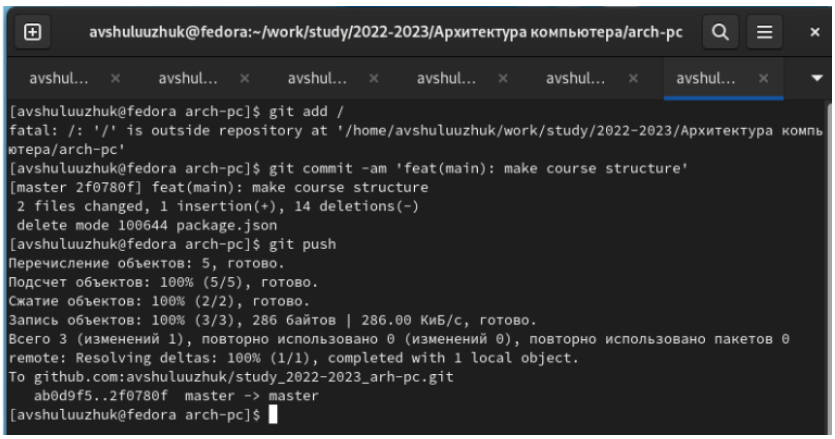

Рис. 3.10: отправление файлов на сервер

Далее проверяем правильность создания иерархии рабочего пространства в локальном репозитории и на странице гит

4 Ответы на контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются? Это программное обеспечение для облегчения работы с изменяющейся информацией. VCS позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия. Хранилище (repository), или репозитарий, место хранения всех версий и служебной информации. Commit (¾[трудовой] вклад¿, не переводится) синоним версии; процесс создания новой версии. История – место, где сохраняются 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 и, затем, добавляет свои изменения обратно. Децентрализованные VCS: у каждого пользователя свой вариант (возможно не один) 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? Git это система управления версиями. У Git две основных задачи: первая - хранить информацию о всех изменениях в вашем коде начиная с самой первой строчки, а вторая обеспечение удобства командной работы над кодом

7. Назовите и дайте краткую характеристику командам git. git –version (Проверка версии Git) git init (Инициализировать ваш текущий рабочий каталог как Git-репозиторий) git clone https://www.github.com/username/repo-name (Скопировать существующий удаленный Git-репозиторий) git remote (Просмотреть список текущих удалённых репозиториев Git) git remote -v (Для более подробного вывода) git add my_script.py (Можете указать в команде конкретный файл). git add . (Позволяет охватить все файлы в текущем каталоге, включая файлы, чье имя начинается с точки) git commit -am “Commit message” (Вы можете сжать все индексированные файлы и отправить коммит). git branch (Просмотреть список текущих веток можно с помощью команды branch) git–help (Чтобы узнать больше обо всех доступных параметрах и командах) git push origin master (Передать локальные коммиты в ветку удаленного репозитория).

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

9. Что такое и зачем могут быть нужны ветви (branches)? Ветки нужны, чтобы несколько программистов могли вести работу над одним и тем же проектом или даже файлом одновременно, при этом не мешая друг другу. Кроме того, ветки используются для тестирования экспериментальных функций: чтобы не повредить основному проекту, создается новая ветка специально для экспериментов

10. Как и зачем можно игнорировать некоторые файлы при commit? Игнорируемые файлы это, как правило, артефакты сборки и файлы, генерируемые

машиной из исходных файлов в вашем репозитории, либо файлы, которые по какой-либо иной причине не должны попадать в коммиты.

5 Выводы

Были приобретены практические навыки по работе с системой git. Изучена идеология и примение средств конторя версий. В процессе лабораторной работы выполнено создание рабочего пространсва и нового репозитория курса на основе шаблона, загрузка файлов на github