

# **Операционные системы**

Лабораторная работа №2

*Первоначальная настройка git*

Емельянов Антон НПМбв-01-21

## **Содержание**

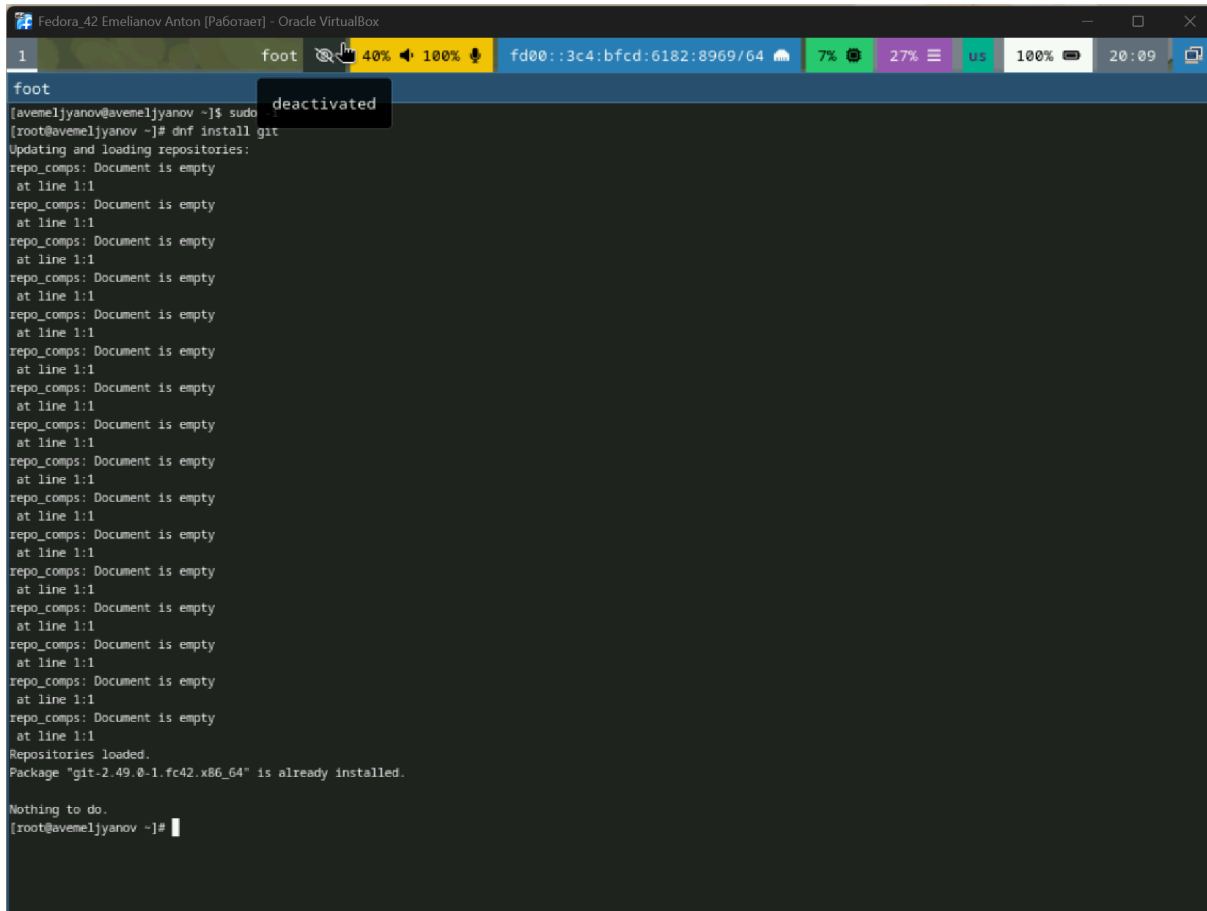
Цель работы	3
Ход работы	4
Вывод	10
Контрольные вопросы	11

## Цель Работы

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

# Ход работы

Начнём установку ПО с git:



```
foot
[avemeljyanov@avemeljyanov ~]$ sudo
[root@avemeljyanov ~]# dnf install git
Updating and loading repositories:
repo_comps: Document is empty
  at line 1:1
repo_comps: Document is empty
  at line 1:1
repo_comps: Document is empty
  at line 1:1
repo_comps: Document is empty
  at line 1:1
repo_comps: Document is empty
  at line 1:1
repo_comps: Document is empty
  at line 1:1
repo_comps: Document is empty
  at line 1:1
repo_comps: Document is empty
  at line 1:1
repo_comps: Document is empty
  at line 1:1
repo_comps: Document is empty
  at line 1:1
repo_comps: Document is empty
  at line 1:1
repo_comps: Document is empty
  at line 1:1
repo_comps: Document is empty
  at line 1:1
repo_comps: Document is empty
  at line 1:1
repo_comps: Document is empty
  at line 1:1
repo_comps: Document is empty
  at line 1:1
repo_comps: Document is empty
  at line 1:1
Repositories loaded.
Package "git-2.49.0-1.fc42.x86_64" is already installed.

Nothing to do.
[root@avemeljyanov ~]#
```

Рис.1 Установка git прошла успешно

Далее установим gh, а также внесём всю базовую информацию: имя глобального пользователя, адрес электронной почты, настроим utf-8 в выводе сообщений git. Зададим имя начальной ветки (master) и установим параметры autocrlf и safecrlf

```
Fedora_42 Emelianov Anton [Pa6opaer] - Oracle VirtualBox
1 foot 40% 100% fd00::3c4:bfc4:6182:8969/64 31% 30% us 100% 20:10
foot
repo_comps: Document is empty
at line 1:1
repo_comps: Document is empty
at line 1:1
repo_comps: Document is empty
at line 1:1
repo_comps: Document is empty
at line 1:1
repo_comps: Document is empty
at line 1:1
repo_comps: Document is empty
at line 1:1
repo_comps: Document is empty
at line 1:1
repo_comps: Document is empty
at line 1:1
repo_comps: Document is empty
at line 1:1
repo_comps: Document is empty
at line 1:1
repo_comps: Document is empty
at line 1:1
Repositories loaded.
Package Arch Version Repository Size
Installing:
gh x86_64 2.74.0-1.fc42 updates 38.9 MiB
Transaction Summary:
Installing: 1 package
Total size of inbound packages is 11 MiB. Need to download 11 MiB.
After this operation, 39 MiB extra will be used (install 39 MiB, remove 0 B).
Is this ok [y/N]: y
[1/1] gh-0:2.74.0-1.fc42.x86_64 100% | 2.6 MiB/s | 11.2 MiB | 00m04s
-----
[1/1] Total 100% | 2.2 MiB/s | 11.2 MiB | 00m05s
Running transaction
[1/3] Verify package files 100% | 28.0 B/s | 1.0 B | 00m00s
[2/3] Prepare transaction 100% | 1.0 B/s | 1.0 B | 00m01s
[3/3] Installing gh-0:2.74.0-1.fc42.x86_64 100% | 18.8 MiB/s | 39.1 MiB | 00m02s
Complete!
[root@emelianov ~]#
```

Рис.2 Установка gh прошла успешно

```
[1/3] Verify package files 100% | 28.0 B/s | 1.0 B | 00m00s
[2/3] Prepare transaction 100% | 1.0 B/s | 1.0 B | 00m01s
[3/3] Installing gh-0:2.74.0-1.fc42.x86_64 100% | 18.8 MiB/s | 39.1 MiB | 00m02s
Complete!
[root@emelianov ~]# git config --global user.name "Anton Emelianov"
[root@emelianov ~]# git config --global user.email "profanton97@gmail.com"
[root@emelianov ~]# git config --global core.quotepath false
[root@emelianov ~]# git config --global init.defaultBranch master
[root@emelianov ~]# git config --global core.autocrlf input
[root@emelianov ~]# git config --global core.safecrlf warn
[root@emelianov ~]#
```

Рис.3 Внесли все базовые параметры.

Сгенерируем ключи размера 4096 по алгоритму RSA и по алгоритму ed25519, также сгенерируем ргр ключ, он понадобится для дальней привязки github. Заодно не забудем зарегистрироваться на github (у меня уже была регистрация, но в данном случае это почти ничего не меняет, так как делаю я с новой виртуальной машины).

```

[root@avemeljyanov ~]# ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): /root/.ssh/id_rsa
Enter passphrase for "/root/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:QU6BpZuArepfQ+mkat1CtyT6dvT0vWp2Ci+YF0/cP6U root@avemeljyanov
The key's randomart image is:
+---[RSA 4096]-----+
|      O=.      |
|      O  .    |
|      . O . O  |
|      . . O .  |
|      . ++ 5   |
|      . O== O . |
|      . +.O=*   |
|      . O.*.*== |
|      . O+O+ =*+ |
+-----[SHA256]-----+
[root@avemeljyanov ~]# ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519): /root/.ssh/id_ed25519
Enter passphrase for "/root/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:Ywq20bKvYTGaQjLN7hK00AS2oiGQ3FBJ7vcPgpEJf18 root@avemeljyanov
The key's randomart image is:
+--[ED25519 256]--+
|+==O.          |
|+.+.O          |
|+=.            |
|*+= O.         |
|*O+*B..ES      |
|+++.+=0oo .    |
|..+.+=O.O      |
|O . O. O       |
| .. ...        |
+-----[SHA256]-----+
[root@avemeljyanov ~]#

```

Рис.4 SSH ключи сгенерированы по алгоритмам RSA и ed25519.

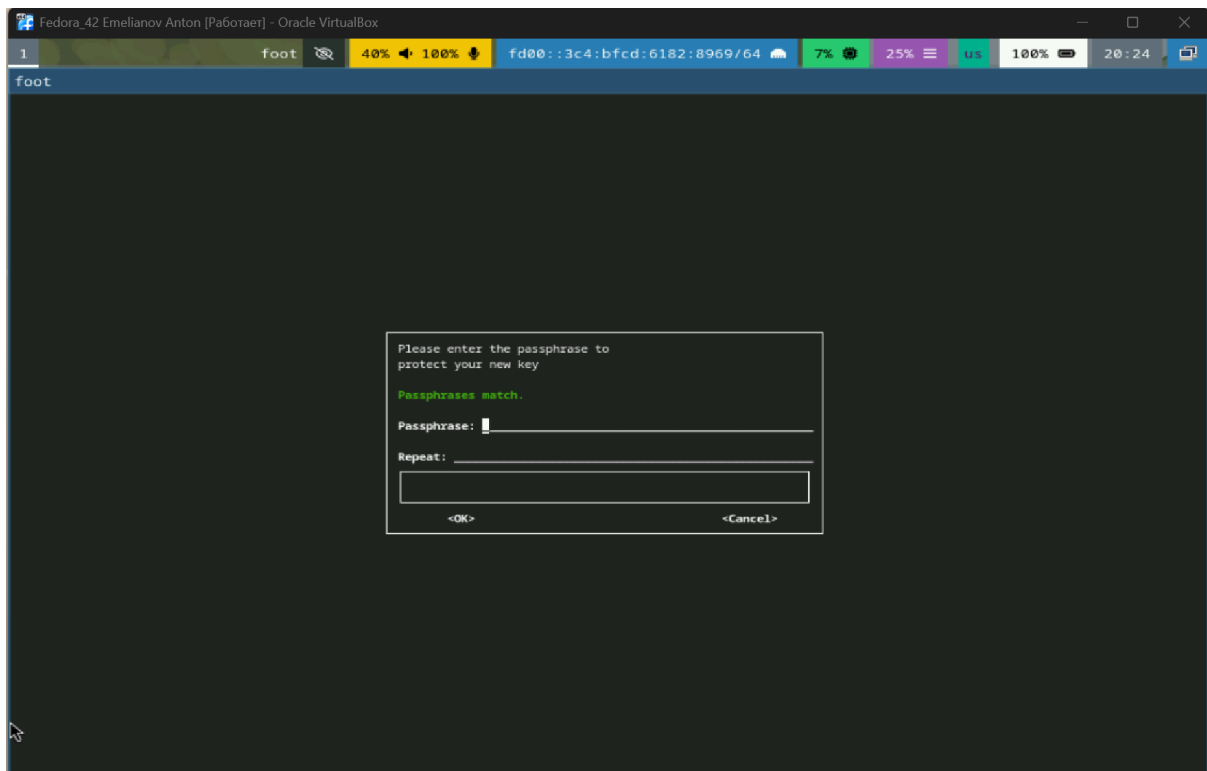


Рис.5 Предложили защитить всё паролем.

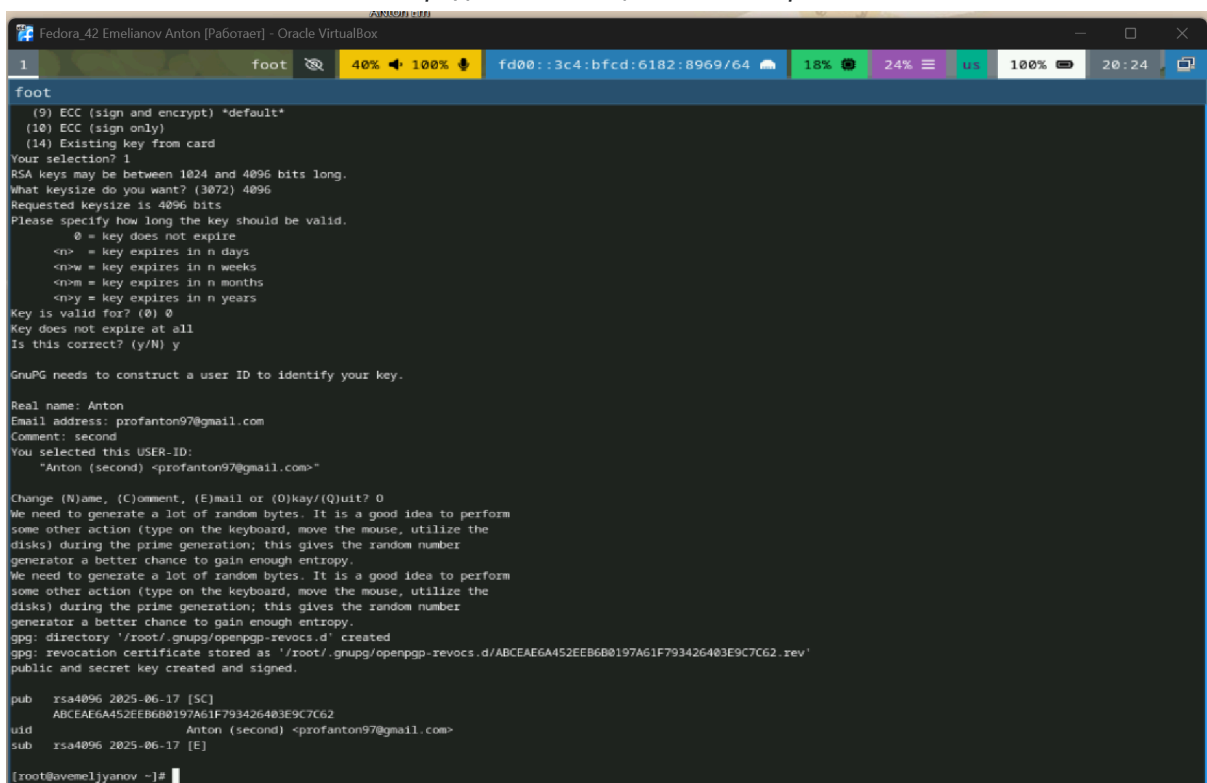


Рис.6 Сгенерирован ргр ключ.

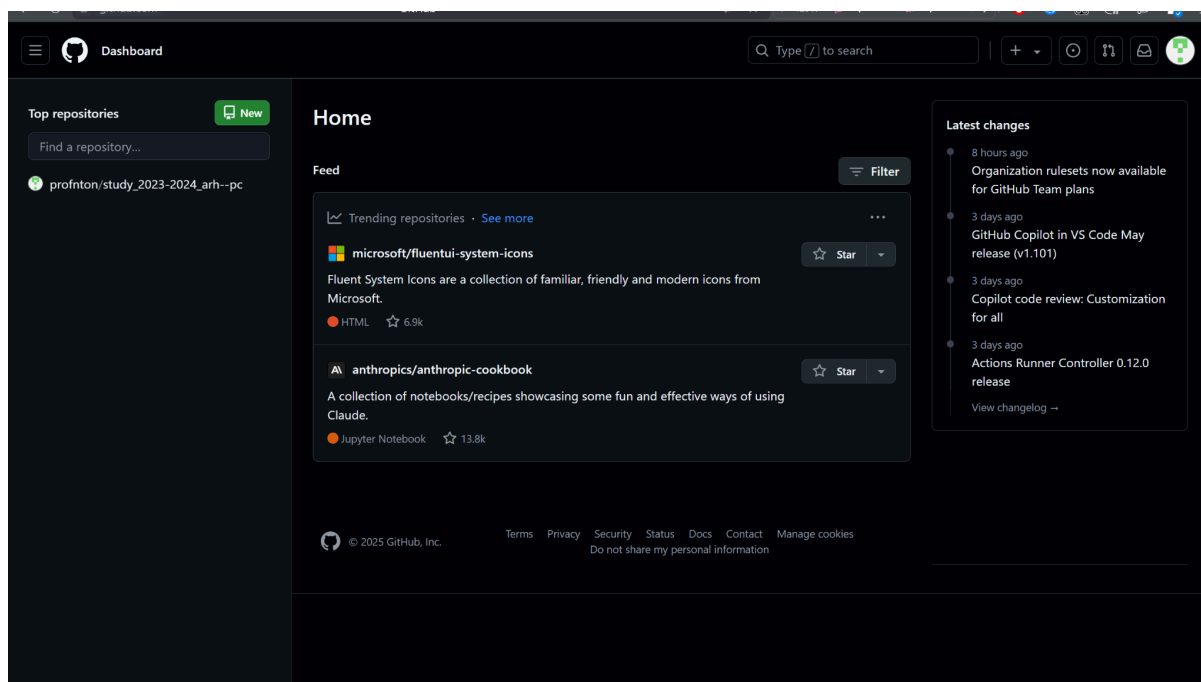


Рис.7 Github “Заведён” и готов к использованию

Теперь добавили ключ на Github, таким образом подключили систему к Github.

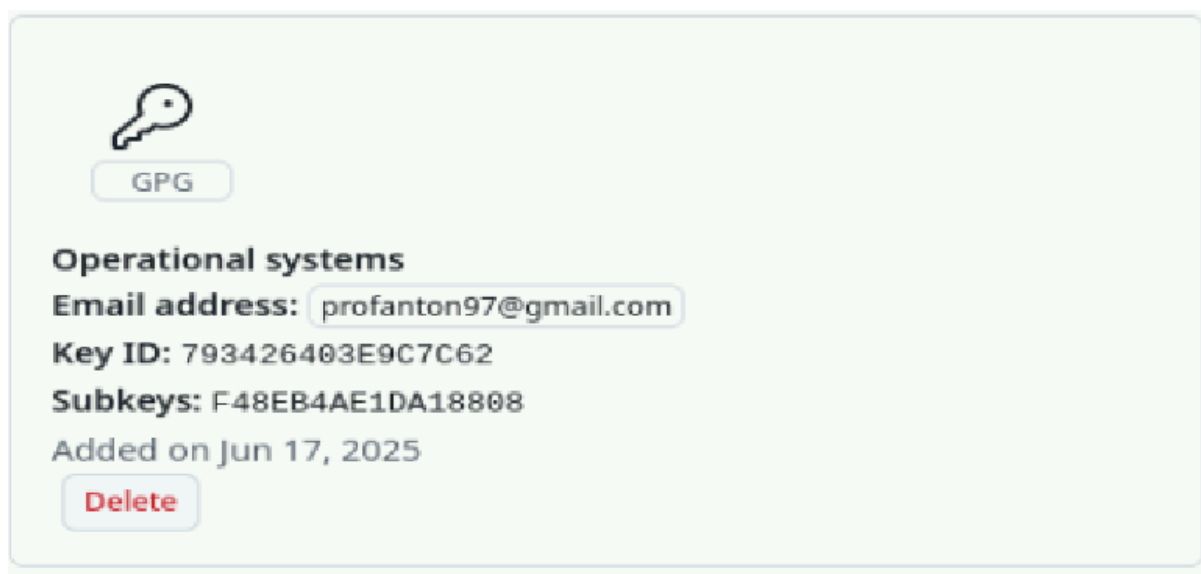


Рис.8 Ключик добавлен

Настроим автоматические подписи коммитов Git, заставим Git использовать мой email для подписи коммитов.

```
[avemeljyanov@avemeljyanov ~]$ git config --global user.signingkey 793426403E9C7C62
[avemeljyanov@avemeljyanov ~]$ git config --global commit.gpgsign true
[avemeljyanov@avemeljyanov ~]$ git config --global gpg.program $(which gpg2)
[avemeljyanov@avemeljyanov ~]$
```

Рис.9 Теперь должен правильно подписывать commit-ы

Произведём авторизацию и создадим шаблон рабочего пространства, для этого создадим папку и в ней образуем репозиторий, скопируем репозиторий и, перейдя в каталог курса, удалим всё лишнее и создадим каталоги.



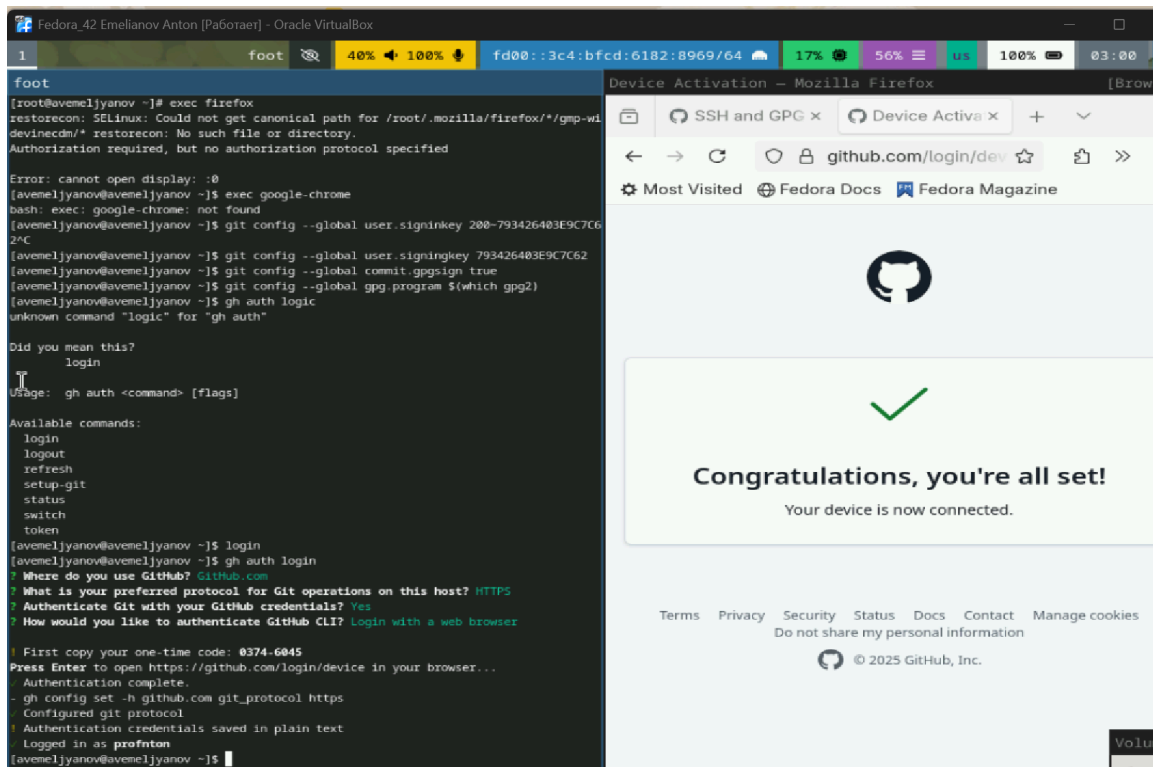


Рис.10 Подрубаем github.

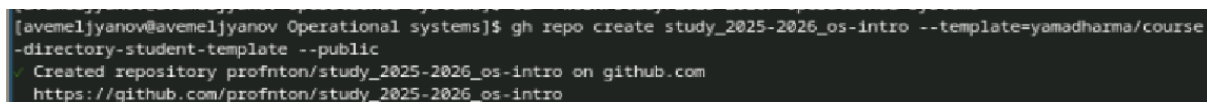


Рис.11 Создаём репозиторий курса.

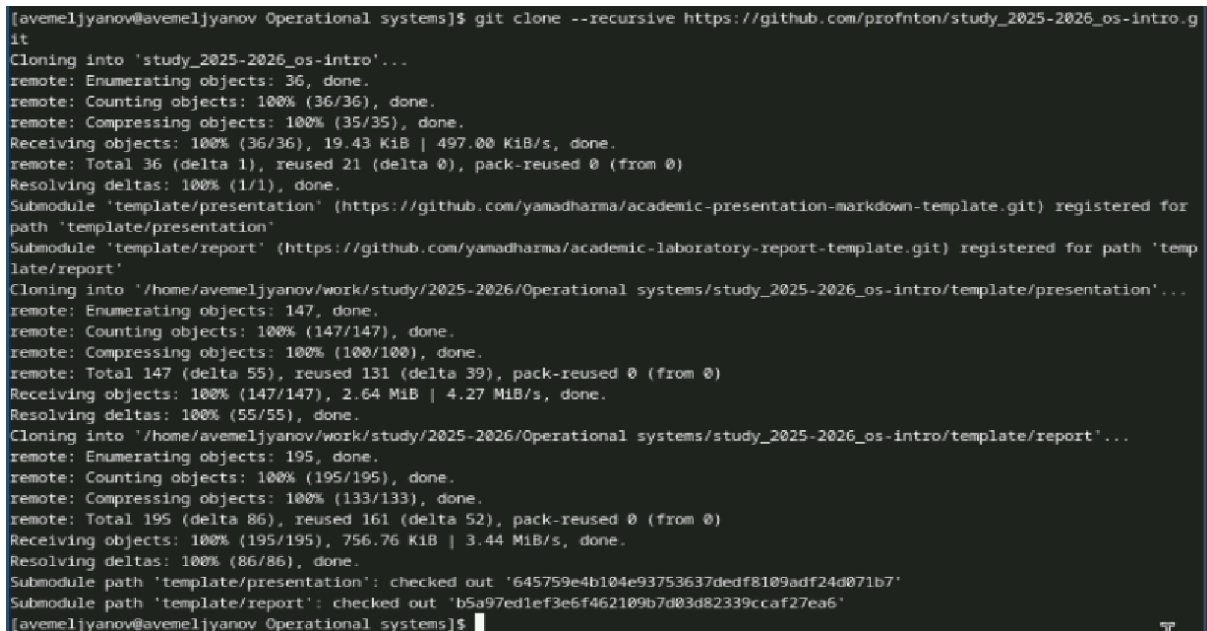


Рис.12 Копируем всё и кое-что удаляем

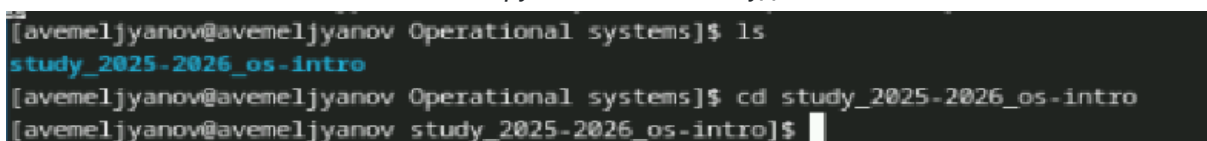


Рис.13 Переходим в каталог курса и готовимся к финальному рывку

Затем отправим файлы на сервер.

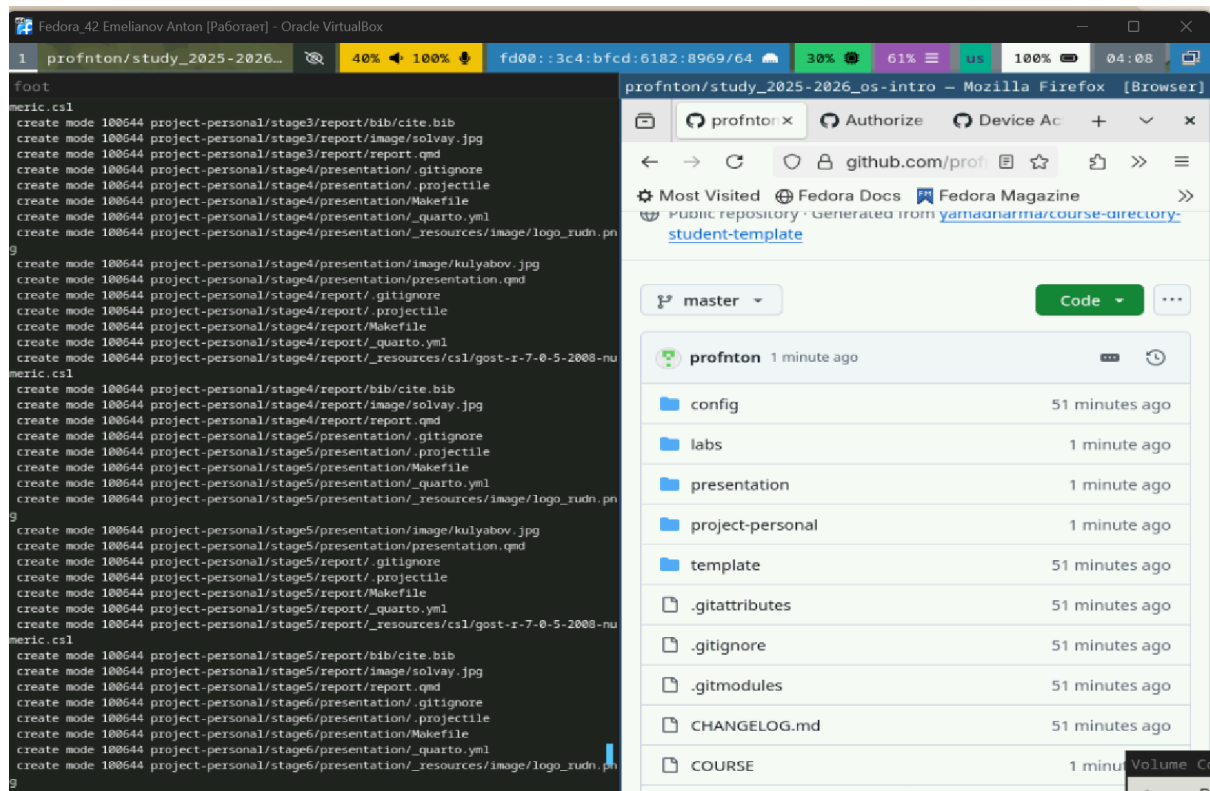


Рис.14 Подтверждаем и отправляем всё. Изменения отразились, УРА

## Вывод

В этой лабораторной работе мы успешно создали и настроили систему контроля версий Git, её окружение и организовали её взаимодействие с Github.

## Контрольные вопросы

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

VCS используется для контроля изменения файлов во времени, их основная задача - контроль версий, сравнение, возврат, совместная работа, ветвление и резервное копирование

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

Хранилище = все версии проекта, а также история их изменений

Commit = снимок состояния проекта в момент времени

История = последовательность Commit-ов

Рабочая копия = локальная копия для работы

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

Централизованные (SVN, CVS) - единое хранилище

Децентрализованные (Git, Mercurial) - у каждого своя полная копия

Опишите действия с VCS при единоличной работе с хранилищем.

Создать/клонировать -> Изменить -> Проверить -> Добавить в индекс -> Закоммитить -> Push (для удалённого)

Опишите порядок работы с общим хранилищем VCS.

Клонировать -> создать ветку -> Изменить -> Проверить -> Добавить в индекс -> Закоммитить -> Push -> Pull request -> Ревью -> Слияние -> Удалить ветку -> Pull

Каковы основные задачи, решаемые инструментальным средством git?

Версионный контроль, совместная работа, управление ветками, отслеживание изменений, разрешение конфликтов, возврат к версиям

Назовите и дайте краткую характеристику командам git.

git init = создать локальный репозиторий

git clone = скачать репозиторий

git add = добавить в индекс сейчас

git commit = зафиксировать изменения локально

git push = отправить на сервер

git pull = получить с сервера

git status = узнать текущее состояние

git log = история коммитов

git branch = создать/переключить ветки

git checkout = переключиться между ветками

git merge = объединить ветки

git diff = показать различия

Приведите примеры использования при работе с локальным и удалённым репозиториями.

Локально: git init, git add file, git commit, git log

Удалённо: git clone URL, git push origin main, git pull origin main

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

Независимые пути разработки, эксперименты, разделение задач, параллельная работа

Как и зачем можно игнорировать некоторые файлы при commit?

Конфиденциальные (не хочу показывать), временные (не хочу сохранять) и больших файлов (ресурсы жалко), сокращение размеров репозитория и предотвращение конфликтов