

# **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

**Дисциплина: Операционные системы**

Обрезкова Анастасия Владимировна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
3.1	Системы контроля версий. Общие понятия . . . . .	7
3.2	Примеры использования git . . . . .	8
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Установка программного обеспечения . . . . .	9
<b>5</b>	<b>Выводы</b>	<b>20</b>
	<b>Список литературы</b>	<b>21</b>

## Список иллюстраций

4.1	Аккаунт . . . . .	9
4.2	Базовая настройка . . . . .	10
4.3	Создание ключа . . . . .	10
4.4	Создание ключа . . . . .	10
4.5	Загруженный на GitHub ключ . . . . .	11
4.6	Создание ключа . . . . .	11
4.7	Создание ключа . . . . .	12
4.8	Отпечаток ключа и копирование в буфер обмена . . . . .	12
4.9	Загрузила ключ на GitHub . . . . .	13
4.10	Настройка подписей коммитов . . . . .	13
4.11	Авторизация . . . . .	13
4.12	Создание репозитория . . . . .	14
4.13	Настройка курса . . . . .	14
4.14	Настройка курса . . . . .	14
4.15	РЕзультат . . . . .	15

## **Список таблиц**

# 1 Цель работы

Изучение идеологии и применения средств контроля версий Git, освоение умения по работе с Git

## 2 Задание

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

## 3 Теоретическое введение

### 3.1 Системы контроля версий. Общие понятия

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.

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.

## 3.2 Примеры использования git

1. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` с различными опциями.
2. Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.



## 4 Выполнение лабораторной работы

### 4.1 Установка программного обеспечения

1. В прошлом семестре я уже установила git и gh и создала учетную запись на GitHub. (рис. [4.1])

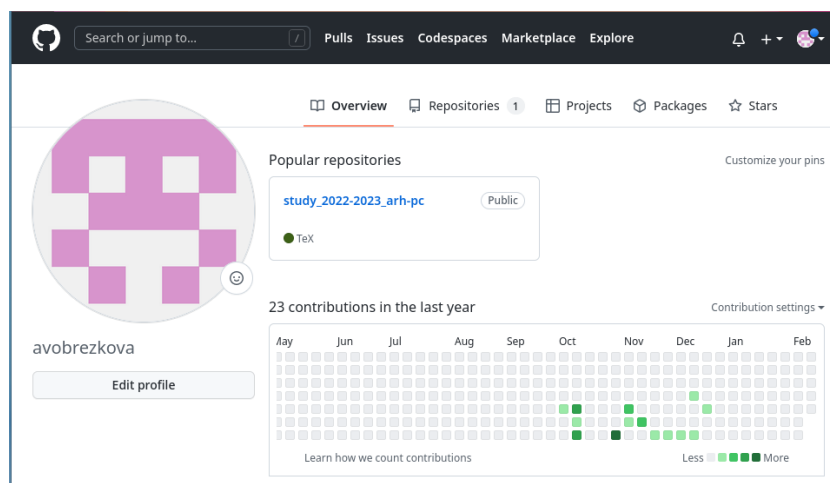
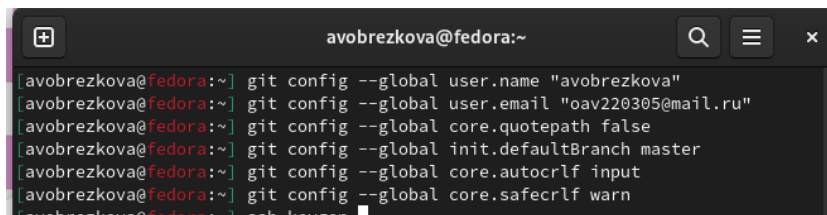


Рис. 4.1: Аккаунт

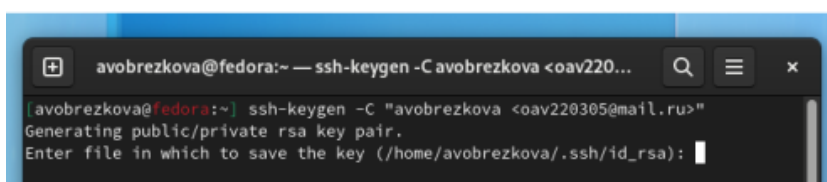
2. Задала базовую настройку git (имя владельца репозитория, настройка utf-8, установка параметров autocrlf, установка параметров safecrlf. (рис. [4.2])

A terminal window titled 'avobrezkova@fedora:~' showing a series of 'git config' commands being executed. The commands are: 'git config --global user.name "avobrezkova"', 'git config --global user.email "oav220305@mail.ru"', 'git config --global core.quotepath false', 'git config --global init.defaultBranch master', 'git config --global core.autocrlf input', and 'git config --global core.safecrlf warn'. The prompt is '[avobrezkova@fedora:~]' for each line.

```
[avobrezkova@fedora:~] git config --global user.name "avobrezkova"
[avobrezkova@fedora:~] git config --global user.email "oav220305@mail.ru"
[avobrezkova@fedora:~] git config --global core.quotepath false
[avobrezkova@fedora:~] git config --global init.defaultBranch master
[avobrezkova@fedora:~] git config --global core.autocrlf input
[avobrezkova@fedora:~] git config --global core.safecrlf warn
[avobrezkova@fedora:~] ssh-keygen
```

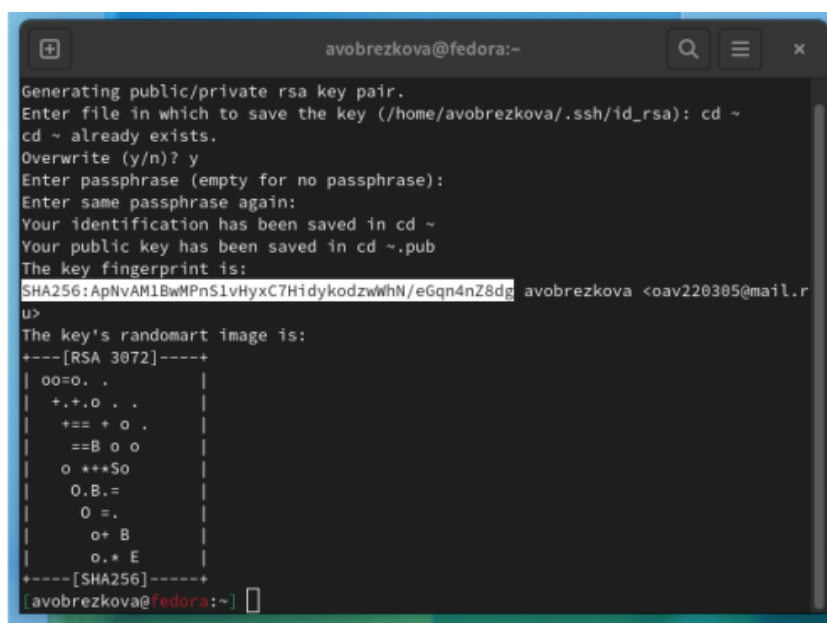
Рис. 4.2: Базовая настройка

3. Создала в прошлом семестре ssh ключ, загрузила его в GitHub. (рис. [4.3], рис. [4.4]; рис. [4.5])

A terminal window titled 'avobrezkova@fedora:~ — ssh-keygen -C avobrezkova <oav220305@mail.ru>' showing the command 'ssh-keygen -C "avobrezkova <oav220305@mail.ru>"' being executed. The output is 'Generating public/private rsa key pair.' and 'Enter file in which to save the key (/home/avobrezkova/.ssh/id\_rsa):'.

```
avobrezkova@fedora:~ — ssh-keygen -C avobrezkova <oav220305@mail.ru>
[avobrezkova@fedora:~] ssh-keygen -C "avobrezkova <oav220305@mail.ru>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/avobrezkova/.ssh/id_rsa):
```

Рис. 4.3: Создание ключа

A terminal window titled 'avobrezkova@fedora:~' showing the completion of the 'ssh-keygen' command. The output includes prompts for passphrase and overwrite, followed by confirmation messages and the key fingerprint: 'SHA256:ApNvAM1BwMPnS1vHyxC7HidykodzwwhN/eGqn4nZ8dg avobrezkova <oav220305@mail.ru>'. It also displays a 'randomart image' for the RSA 3072 key.

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/avobrezkova/.ssh/id_rsa): cd ~
cd ~ already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in cd ~
Your public key has been saved in cd ~/.pub
The key fingerprint is:
SHA256:ApNvAM1BwMPnS1vHyxC7HidykodzwwhN/eGqn4nZ8dg avobrezkova <oav220305@mail.ru>
The key's randomart image is:
+---[RSA 3072]-----+
| oo=O. . |
| +.+.. |
| +== + o |
| ==B o o |
| o +++So |
| O.B.= |
| O =. |
| o+ B |
| o.* E |
+----[SHA256]-----+
[avobrezkova@fedora:~]
```

Рис. 4.4: Создание ключа

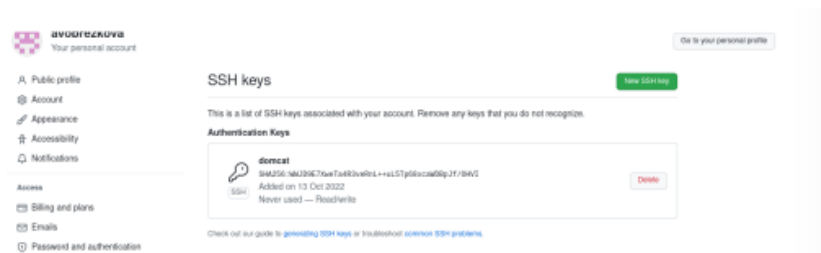


Рис. 4.5: Загруженный на GitHub ключ

#### 4. Создание gpg ключа. (рис. [4.6]; рис. [4.7])

```
[avobrezkova@fedora:~] gpg --full-generate-key
gpg (GnuPG) 2.3.4; Copyright (C) 2021 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: создан каталог '/home/avobrezkova/.gnupg'
gpg: создан щит с ключами '/home/avobrezkova/.gnupg/pubring.kbx'
Выберите тип ключа:
  (1) RSA and RSA
  (2) DSA and Elgamal
  (3) DSA (sign only)
  (4) RSA (sign only)
  (9) ECC (sign and encrypt) *default*
  (10) ECC (только для подписи)
  (14) Existing key from card
Ваш выбор? 1
длина ключей RSA может быть от 1024 до 4096.
Какой размер ключа Вам необходим? (3072) 4096
Запрошенный размер ключа - 4096 бит
Выберите срок действия ключа.
  0 = не ограничен
  <n> = срок действия ключа - n дней
  <n>w = срок действия ключа - n недель
  <n>m = срок действия ключа - n месяцев
  <n>y = срок действия ключа - n лет
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y
```

Рис. 4.6: Создание ключа

```

Ваше полное имя: avobrezkova
Адрес электронной почты: oav220305@mail.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
"avobrezkova <oav220305@mail.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? 0
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
Необходимо получить много случайных чисел. Желательно, чтобы Вы
в процессе генерации выполняли какие-то другие действия (печать
на клавиатуре, движения мыши, обращения к дискам); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: /home/avobrezkova/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/avobrezkova/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/avobrezkova/.gnupg/openpgp-revocs.d/166147B66E10CE547B2A8B
8D5616185C60137B12.rev'.
открытый и секретный ключи созданы и подписаны.

pub  rsa4096 2023-02-16 [SC]
    166147B66E10CE547B2A8B8D5616185C60137B12
uid          avobrezkova <oav220305@mail.ru>
sub  rsa4096 2023-02-16 [E]

[avobrezkova@fedora:~]

```

Рис. 4.7: Создание ключа

5. Вывила список ключей и скопировала отпечаток приватного ключа, скопировала сгенерированный ключ в буфер обмена и добавила его на GitHub. (рис. [4.8], рис. [4.9])

```

[avobrezkova@fedora:~] gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3 completes needed: 1 trust model: pgp
gpg: глубина: 0 достоверных: 1 подписанных: 0 доверие: 0-, 0q, 0n, 0m, 0f, 1u
/home/avobrezkova/.gnupg/pubring.kbx
-----
sec  rsa4096/5616185C60137B12 2023-02-16 [SC]
    166147B66E10CE547B2A8B8D5616185C60137B12
uid          [ абсолютно ] avobrezkova <oav220305@mail.ru>
ssb  rsa4096/42187E8139C2254C 2023-02-16 [E]

[avobrezkova@fedora:~] gpg --armor --export <PGP Fingerprint> | xclip -sel clip
bash: синтаксическая ошибка рядом с неожиданным маркером «|»
[avobrezkova@fedora:~] gpg --armor --export 5616185C60137B12 | xclip -sel clip
[avobrezkova@fedora:~]


```

Рис. 4.8: Отпечаток ключа и копирование в буфер обмена

## GPG keys

[New GPG key](#)

This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.

 **Email address:** oav220305@mail.ru  
**Key ID:** 5616185C60137B12  
**Subkeys:** 42187E8139C2254C  
Added on Feb 16, 2023

[Delete](#)

Рис. 4.9: Загрузила ключ на GitHub

### 6. Настройка автоматических подписей коммитов git. (рис. [4.10])

```
avobrezkova@fedora:~$ git config --global user.signingkey 5616185C60137B12
avobrezkova@fedora:~$ git config --global commit.gpgsign true
avobrezkova@fedora:~$ git config --global gpg.program
avobrezkova@fedora:~$
```

Рис. 4.10: Настройка подписей коммитов

### 7. Авторизировалась и настроила gh. (рис. [4.11])

```
avobrezkova@fedora:~/work/study/2022-2023/Операционные системы$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: AAAE-2003
Press Enter to open github.com in your browser...

a/ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as avobrezkova
```

Рис. 4.11: Авторизация

### 8. Создала репозиторий курса на основе шаблона. (рис. [4.12])

```

avobrezkova@fedora:~/work/study/2022-2023/Операционные системы$ mkdir -p ./work/study/2022-2023/"Операционные Системы"
avobrezkova@fedora:~/work/study/2022-2023/Операционные системы$ cd ./work/study/2022-2023/"Операционные системы"
avobrezkova@fedora:~/work/study/2022-2023/Операционные системы$ gh repo create study_2022-2023_os-intro --template=yamadharma/course-directory-student-template
--public
Created repository avobrezkova/study_2022-2023_os-intro on GitHub
avobrezkova@fedora:~/work/study/2022-2023/Операционные системы$ git clone --recursive git@github.com:avobrezkova/study_2022-2023_os-intro.git os-intro
Клонирование в «os-intro»...
remote: Enumerating objects: 27, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 27 (delta 1), reused 11 (delta 0), pack-reused 0
Получение объектов: 100% (27/27), 15.93 KiB | 1.13 MiB/c, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharma/academic-presentation-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharma/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/avobrezkova/work/study/2022-2023/Операционные системы/os-intro/template/presentation»...
remote: Enumerating objects: 82, done.
remote: Counting objects: 100% (82/82), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 82 (delta 28), reused 77 (delta 23), pack-reused 0
Получение объектов: 100% (82/82), 92.90 KiB | 251.80 KiB/c, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/home/avobrezkova/work/study/2022-2023/Операционные системы/os-intro/template/report»...
remote: Enumerating objects: 101, done.
remote: Counting objects: 100% (101/101), done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 101 (delta 40), reused 88 (delta 27), pack-reused 0
Получение объектов: 100% (101/101), 327.25 KiB | 414.80 KiB/c, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be3800ee91f5809264cb755d316174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a8321f0b2e2aeef1a33b4e3b2'
avobrezkova@fedora:~/work/study/2022-2023/Операционные системы$

```

Рис. 4.12: Создание репозитория

9. Настроила каталог курса, удалила лишние файлы, создала каталоги и отправила файлы на сервер. (рис. [4.13]; рис. [4.14]; рис. [4.15])

```

avobrezkova@fedora:~/work/study/2022-2023/Операционные системы$ cd ~/work/study/2022-2023/"Операционные системы"/os-intro
avobrezkova@fedora:~/work/study/2022-2023/Операционные системы/os-intro$ rm package.json
avobrezkova@fedora:~/work/study/2022-2023/Операционные системы/os-intro$ echo os-intro > COURSE
avobrezkova@fedora:~/work/study/2022-2023/Операционные системы/os-intro$ make
avobrezkova@fedora:~/work/study/2022-2023/Операционные системы/os-intro$ git add .
avobrezkova@fedora:~/work/study/2022-2023/Операционные системы/os-intro$ git commit -am 'feat(main): make course structure'
(master => f4fb311) feat(main): make course structure
361 files changed, 100327 insertions(+), 14 deletions(-)
create mode 100644 labs/README.md
create mode 100644 labs/README.rund
create mode 100644 labs/lab01/presentation/Makefile
create mode 100644 labs/lab01/presentation/image/kulyabov.jpg
create mode 100644 labs/lab01/presentation/presentation.md
create mode 100644 labs/lab01/report/Makefile
create mode 100644 labs/lab01/report/bib/cite.bib
create mode 100644 labs/lab01/report/image/placing_800_600_tech.jpg
create mode 100644 labs/lab01/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_fignos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_secnos.py
create mode 100755 labs/lab01/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/_init_.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/core.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/main.py

```

Рис. 4.13: Настройка курса

```

create mode 100644 project-personal/stage0/presentation/image/kulyabov.jpg
create mode 100644 project-personal/stage0/presentation/presentation.md
create mode 100644 project-personal/stage0/report/Makefile
create mode 100644 project-personal/stage0/report/bib/cite.bib
create mode 100644 project-personal/stage0/report/image/placing_800_600_tech.jpg
create mode 100644 project-personal/stage0/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 project-personal/stage0/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 project-personal/stage0/report/pandoc/filters/pandoc_fignos.py
create mode 100755 project-personal/stage0/report/pandoc/filters/pandoc_secnos.py
create mode 100755 project-personal/stage0/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 project-personal/stage0/report/pandoc/filters/pandocxnos/_init_.py
create mode 100644 project-personal/stage0/report/pandoc/filters/pandocxnos/core.py
create mode 100644 project-personal/stage0/report/pandoc/filters/pandocxnos/main.py
create mode 100644 project-personal/stage0/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 project-personal/stage0/report/report.md
avobrezkova@fedora:~/work/study/2022-2023/Операционные системы/os-intro$ git push
Перечисление объектов: 40, готово.
Посчет объектов: 100% (40/40), готово.
Сжатие объектов: 100% (38/38), готово.
Пакеты объектов: 100% (38/38), 243.04 KiB | 2.02 MiB/c, готово.
Всего 38 (изменений 4), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:avobrezkova/study_2022-2023_os-intro.git
  f4fb311..84fb311 master -> master
avobrezkova@fedora:~/work/study/2022-2023/Операционные системы/os-intro$

```

Рис. 4.14: Настройка курса

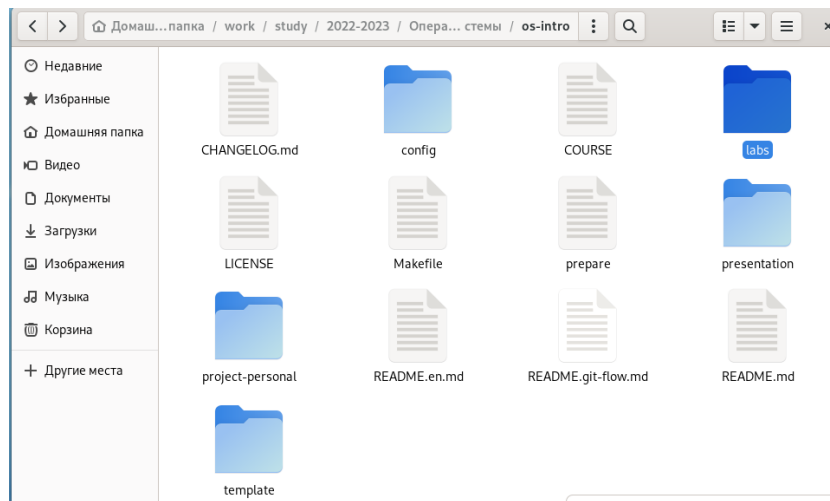


Рис. 4.15: РЕзультат

Данные изменения можно проверить по ссылке: [https://github.com/avobrezkova/study\\_2022-2023\\_os-intro/tree/master/labs/lab02](https://github.com/avobrezkova/study_2022-2023_os-intro/tree/master/labs/lab02)

#### #Контрольные вопросы

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

Системы контроля версий — это программные инструменты, помогающие командам разработчиков управлять изменениями в исходном коде с течением времени. Какие задачи решает система контроля версий:

- Защищает исходный код от потери. Данные хранятся на удалённом сервере, даже если разработчики удалят файлы с локального компьютера, они останутся в репозитории.

- Обеспечивает командную работу.
- Помогает отменить изменения.
- Распределённая работа.

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

Хранилище (repository, сокр. repo), или репозиторий, — место хранения всех версий и служебной информации.

Коммит (commit; редко переводится как «слепок») — 1) синоним версии; 2) создание новой версии («сделать коммит», «закоммитить»).

Рабочая копия (working copy или working tree) — текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней)

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

Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Пример - Wikipedia.

В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Пример — Bitcoin.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером.

### 4. Опишите действия с VCS при единоличной работе с хранилищем.

Создадим локальный репозиторий. Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория:

```
git config --global user.name "Имя Фамилия".
```

```
git config --global user.email "work@mail"
```

и настроив utf-8 в выводе сообщений git:

```
git config --global core.quotePath false
```

Для инициализации локального репозитория, расположенного, например, в каталоге ~/tutorial, необходимо ввести в командной строке:



```
cd
mkdir tutorial
cd tutorial
git init
```

5. Опишите порядок работы с общим хранилищем VCS.

Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый): `ssh-keygen -C "Имя Фамилия work@mail"` Ключи сохраняются в каталоге `~/.ssh/`. Скопировав из локальной консоли ключ в буфер обмена

```
cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

вставляем ключ в появившееся на сайте поле.

6. Каковы основные задачи, решаемые инструментальным средством git?

У Git две основных задачи:

хранить информацию о всех изменениях в вашем коде, начиная с самой первой строки,

обеспечение удобства командной работы над кодом.

7. Назовите и дайте краткую характеристику командам git.

Наиболее часто используемые команды

- `git`: – создание основного дерева репозитория:
- `git init`–получение обновлений (изменений)текущего дерева из центрального репозитория:
- `git pull`–отправка всех произведённых изменений локального дерева в центральный репозиторий:
- `git push`–просмотр списка изменённых файлов втекущей директории:`git status`–просмотртекущих изменения:
- `git diff`–сохранениетекущих изменений:–добавить все изменённые и/или созданные файлы и/или каталоги:

- `git add .` – добавить конкретные изменённые и/или созданные файлы и/или каталоги:

- `git add имена_файлов` – удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории):

- `git rm имена_файлов` – сохранение добавленных изменений: – сохранить все добавленные изменения и все изменённые файлы:

- `git commit -am 'Описание коммита'` – сохранить добавленные изменения с внесением комментария через встроенный редактор:

- `git commit` – создание новой ветки, базирующейся на текущей:

- `git checkout -b имя_ветки` – переключение на некоторую ветку:

- `git checkout имя_ветки` (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) – отправка изменений конкретной ветки в центральный репозиторий:

- `git push origin имя_ветки` – слияние ветки стекущим деревом:

- `git merge --no-ff имя_ветки` – удаление ветки: – удаление локальной уже слитой с основным деревом ветки:

- `git branch -d имя_ветки` – принудительное удаление локальной ветки:

- `git branch -D имя_ветки` – удаление ветки с центрального репозитория:

- `git push origin :имя_ветки`

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

Использование `git` при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий):

```
git add hello.txt
```

```
git commit -am 'Новый файл'
```

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

Ветки очень облегчают работу. Они решают такие проблемы как: нужно постоянно создавать архивы с рабочим кодом сложно “переключаться” между архива-

ми сложно перетаскивать изменения между архивами легко что-то напутать или потерять

#### 10. Как и зачем можно игнорировать некоторые файлы при commit?

Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых придобавлении в репозиторий типов файлов в файл .gitignore с помощью сервисов. Для этого сначала нужно получить список имеющихся шаблонов:

```
curl -L -s https://www.gitignore.io/api/list
```

Затем скачать шаблон, например, для C и C++

```
curl -L -s https://www.gitignore.io/api/c » .gitignore
```

```
curl -L -s https://www.gitignore.io/api/c++ » .gitignore
```

## 5 Выводы

В ходе лабораторной работы я изучила идеологию и применение средств контроля версий git, а также освоила умения по работе с git.

## **Список литературы**

1. <https://esystem.rudn.ru/mod/page/view.php?id=971076>