

Отчет по лабораторной работе №2

Первоначальная настройка git

Федорова Наталия Артемовна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Контрольные вопросы	19

Список иллюстраций

4.1	Установка git	8
4.2	Установка gh	9
4.3	Базовая настройка git	10
4.4	Создание ssh ключа	11
4.5	Создание pgp ключа	12
4.6	Учетная запись GitHub	13
4.7	Добавление ключей	14
4.8	Настройка коммитов	14
4.9	Авторизация	15
4.10	Авторизация	15
4.11	Создание репозитория	16
4.12	Отправка на сервер	17
4.13	Отправка на сервер	18

Список таблиц

1 Цель работы

Научиться оформлять отчёты с помощью легковесного языка разметки Markdown.

2 Задание

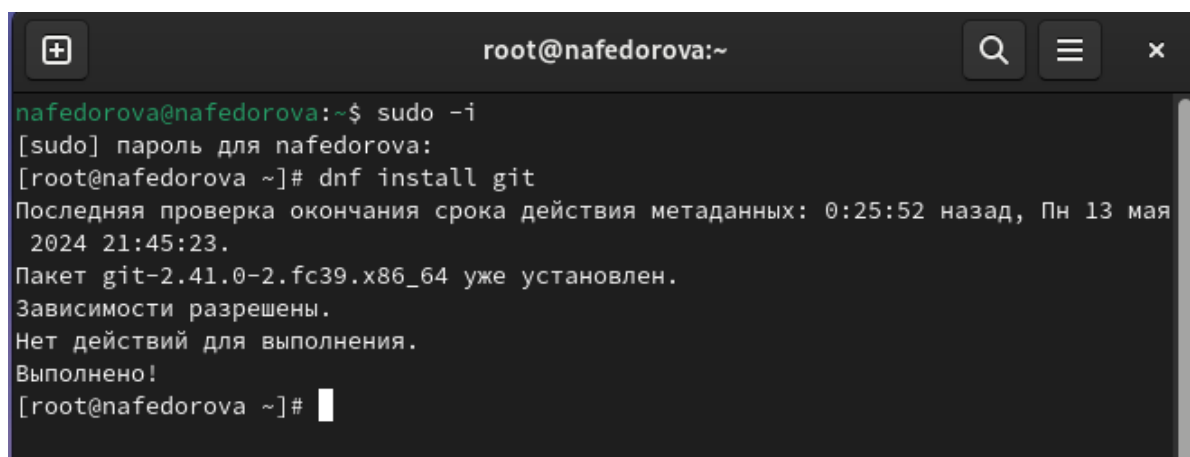
1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Настроить подписи git.
5. Зарегистрироваться на Github.
6. Создать локальный каталог для выполнения заданий по предмету.

3 Теоретическое введение

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. Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом. Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

4 Выполнение лабораторной работы

Для установки **git** надо переключиться на роль супер-пользователя с помощью команды **sudo -i**, после вводим команду **dnf install git**. У меня git уже установлен(рис. 4.1).

A screenshot of a terminal window titled 'root@nafedorova:~'. The terminal shows the following commands and output:

```
nafedorova@nafedorova:~$ sudo -i
[sudo] пароль для nafedorova:
[root@nafedorova ~]# dnf install git
Последняя проверка окончания срока действия метаданных: 0:25:52 назад, Пн 13 мая 2024 21:45:23.
Пакет git-2.41.0-2.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
[root@nafedorova ~]#
```

Рис. 4.1: Установка git

Для установки **gh** вводим команду **dnf install**. Соглашаюсь с установкой и жду окончания процесса (рис. 4.2).


```
root@nafedorova:~  
[root@nafedorova ~]# dnf install gh  
Последняя проверка окончания срока действия метаданных: 0:27:00 назад, Пн 13 мая 2024  
21:45:23.  
Зависимости разрешены.  
=====
```

Пакет	Архитектура	Версия	Репозиторий	Размер
Установка:				
gh	x86_64	2.45.0-1.fc39	updates	9.1 М

```
=====
```

Результат транзакции

Установка 1 Пакет

Объем загрузки: 9.1 М
Объем изменений: 46 М
Продолжить? [д/Н]: д
Загрузка пакетов:

Пакет	Скорость	Объем	Время
gh-2.45.0-1.fc39.x86_64.rpm	8.3 MB/s	9.1 MB	00:01

Общий размер 5.4 MB/s | 9.1 MB 00:01

Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции проведен успешно.
Выполнение транзакции

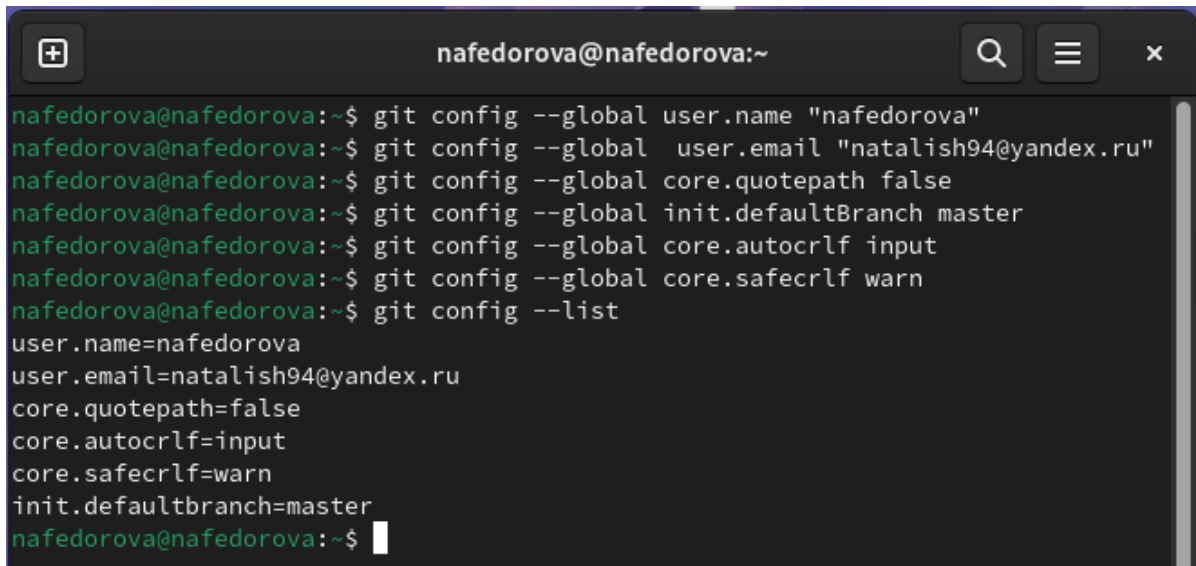
Операция	Пакет	Прогресс
Подготовка		1/1
Установка	gh-2.45.0-1.fc39.x86_64	1/1
Запуск скриптлета	gh-2.45.0-1.fc39.x86_64	1/1
Проверка	gh-2.45.0-1.fc39.x86_64	1/1

Установлен:
gh-2.45.0-1.fc39.x86_64

Выполнено!
[root@nafedorova ~]#

Рис. 4.2: Установка gh

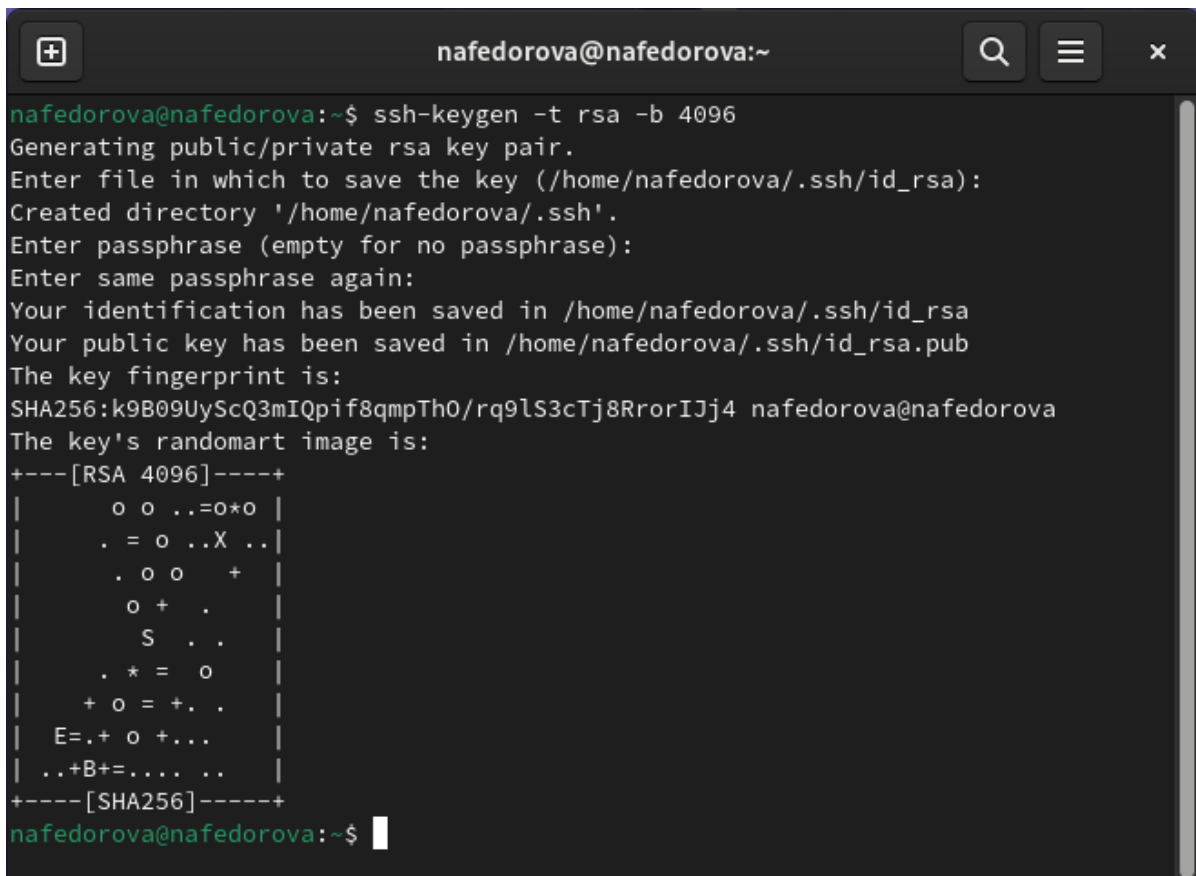
Сделаю базовые настройки git. Для этого задам имя и почту своего репозитория **git config --global user.name "Name Surname"** и **git config --global user.email "work@mail"**, настрою utf-8 в выводе сообщений git **git config --global core.quotePath false**, задам имя начальной ветки master **git config --global init.defaultBranch master** и установлю пару параметров **git config --global core.autocrlf input** и **git config --global core.safecrlf warn**. Проверка изменения с помощью команды **git config --list** (рис. 4.3).

A terminal window titled 'nafedorova@nafedorova:~' with search, menu, and close icons in the title bar. The terminal shows a series of 'git config' commands being executed to set global user and core settings. The output of the 'git config --list' command is displayed at the bottom.

```
nafedorova@nafedorova:~$ git config --global user.name "nafedorova"
nafedorova@nafedorova:~$ git config --global user.email "natalish94@yandex.ru"
nafedorova@nafedorova:~$ git config --global core.quotepath false
nafedorova@nafedorova:~$ git config --global init.defaultBranch master
nafedorova@nafedorova:~$ git config --global core.autocrlf input
nafedorova@nafedorova:~$ git config --global core.safecrlf warn
nafedorova@nafedorova:~$ git config --list
user.name=nafedorova
user.email=natalish94@yandex.ru
core.quotepath=false
core.autocrlf=input
core.safecrlf=warn
init.defaultbranch=master
nafedorova@nafedorova:~$
```

Рис. 4.3: Базовая настройка git

Далее создаю ключ **ssh** по алгоритму **rsa** с ключом размером 4096 бит с помощью команды **ssh-keygen -t rsa -b 4096** (рис. 4.4).

A terminal window titled 'nafedorova@nafedorova:~' with search, menu, and close icons in the title bar. The terminal shows the execution of 'ssh-keygen -t rsa -b 4096'. It prompts for a file path (defaulting to /home/nafedorova/.ssh/id_rsa), creates the directory, asks for a passphrase (left empty), and displays the key fingerprint and a randomart image for the RSA 4096 key. The randomart image is a grid of characters representing the key's fingerprint. The terminal ends with the prompt 'nafedorova@nafedorova:~\$' and a cursor.

```
nafedorova@nafedorova:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/nafedorova/.ssh/id_rsa):
Created directory '/home/nafedorova/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/nafedorova/.ssh/id_rsa
Your public key has been saved in /home/nafedorova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:k9B09UyScQ3mIQpif8qmpTh0/rq9lS3cTj8RrorIJj4 nafedorova@nafedorova
The key's randomart image is:
+----[RSA 4096]-----+
|  o o ..=o*o |
|  . = o ..X ..|
|  . o o  +  |
|  o +  .    |
|  S  . .    |
|  . * = o    |
|  + o = +. .  |
|  E=.+ o +... |
|  ..+B+=.... ..|
+----[SHA256]-----+
nafedorova@nafedorova:~$
```

Рис. 4.4: Создание ssh ключа

Дальше генерирую ключ **pgp** с помощью команды **gpg --full-generate-key**.

Из предложенных опций выбираю тип RSA и RSA, размер 4096 и срок действия 0 (срок действия не ограничен). Так же ввожу личную информацию, которая сохранится в ключе. Важно, чтобы почта соответствовала адресу, используемому на GitHub (рис. 4.5).

```
nafedorova@nafedorova:~$ gpg --full-generate-key
gpg (GnuPG) 2.4.3; Copyright (C) 2023 g10 Code GmbH
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/nafedorova/.gnupg'
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

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: nafedorova
Адрес электронной почты: natalish94@yandex.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
  "nafedorova <natalish94@yandex.ru>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? O
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/nafedorova/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/nafedorova/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/nafedorova/.gnupg/openpgp-revocs.d/4176E4D4
5DB7EC68DA11C70147EB443DA4C90182.rev'.
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2024-05-13 [SC]
      4176E4D45DB7EC68DA11C70147EB443DA4C90182
uid           nafedorova <natalish94@yandex.ru>
sub   rsa4096 2024-05-13 [E]
```

Рис. 4.5: Создание ргр ключа

Далее нужно создать учетную запись на GitHub, но у меня она уже есть. (рис. 4.6).

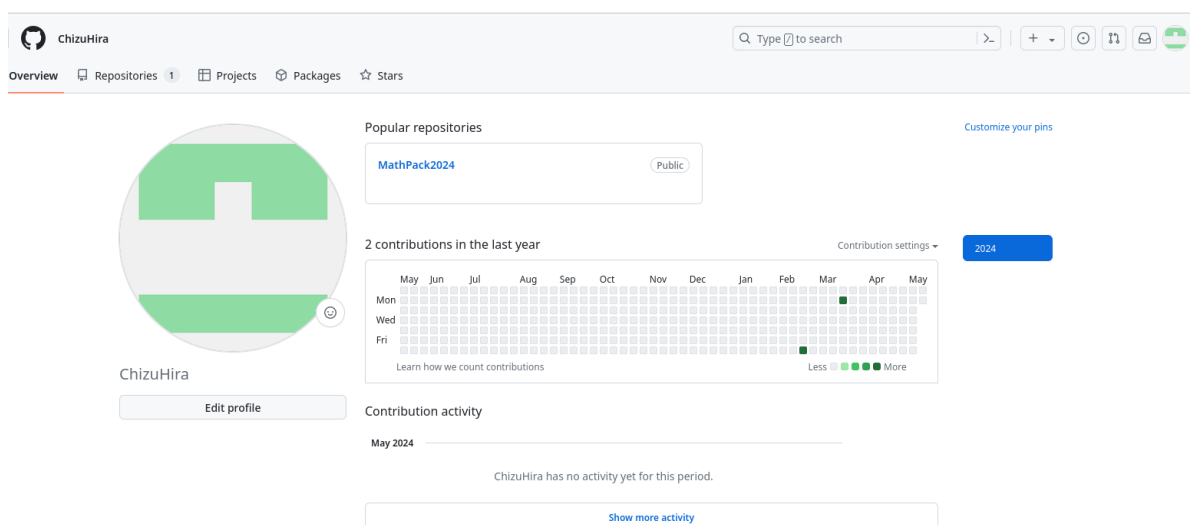


Рис. 4.6: Учетная запись GitHub

Вывожу список ключей и копирую опечаток приватного ключа. Чтобы вывести список используем команду **gpg --list-secret-keys --keyid-format LONG**

Опечаток ключа находится в строке:

sec Алгоритм/Отпечаток_ключа Дата_создания [Флаги] [Годен_до]

Копирую его командой **gpg --armor --export | xclip -sel clip**.

После скопированный ключ добавляю на GitHub (**settings -> SSH and GPG keys -> New GPG key**). Подобным образом добавляю и ssh ключ (рис. 4.7).

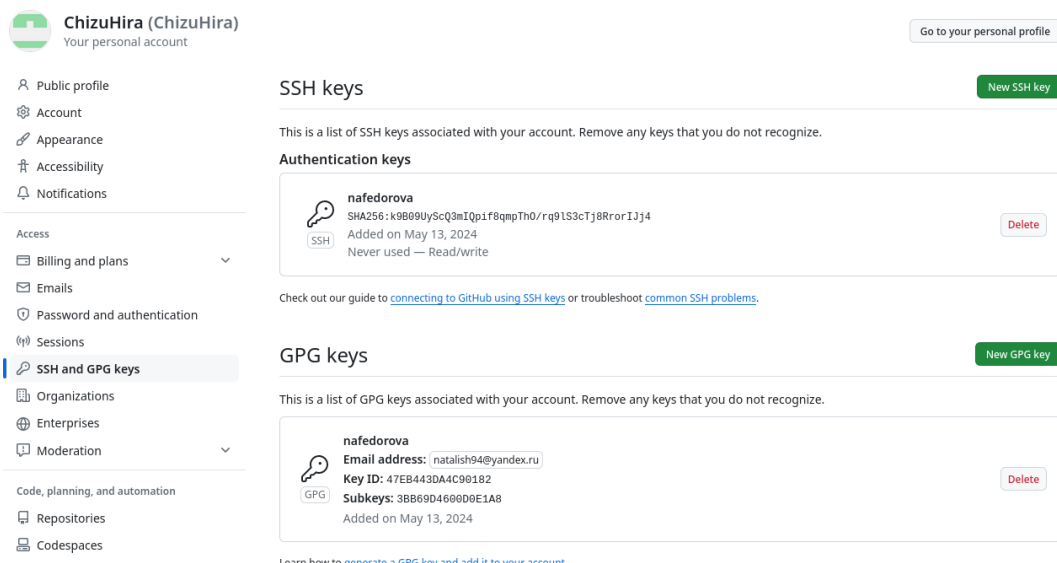


Рис. 4.7: Добавление ключей

Настраиваю автоматические подписи коммитов git с помощью команд: **git config --global user.signingkey** , **git config --global commit.gpgsign true**, **git config --global gpg.program \$(which gpg2)** (рис. 4.8).

```

nafedorova@nafedorova:~$ git config --global user.signingkey 47EB443DA4C90182
nafedorova@nafedorova:~$ git config --global commit.gpgsign true
nafedorova@nafedorova:~$ git config --global gpg.program $(which gpg2)
nafedorova@nafedorova:~$

```

Рис. 4.8: Настройка коммитов

Дальше стоит настроить gh. Для этого ввожу команду **gh auth login**. Авторизируюсь через браузер и ввожу код из терминала (рис. 4.9), (рис. 4.10).

```

nafedorova@nafedorova:~$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? 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: F7D4-9674
Press Enter to open github.com in your browser...
✓ Authentication complete.
- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as ChizuHira
nafedorova@nafedorova:~$

```

Рис. 4.9: Авторизация

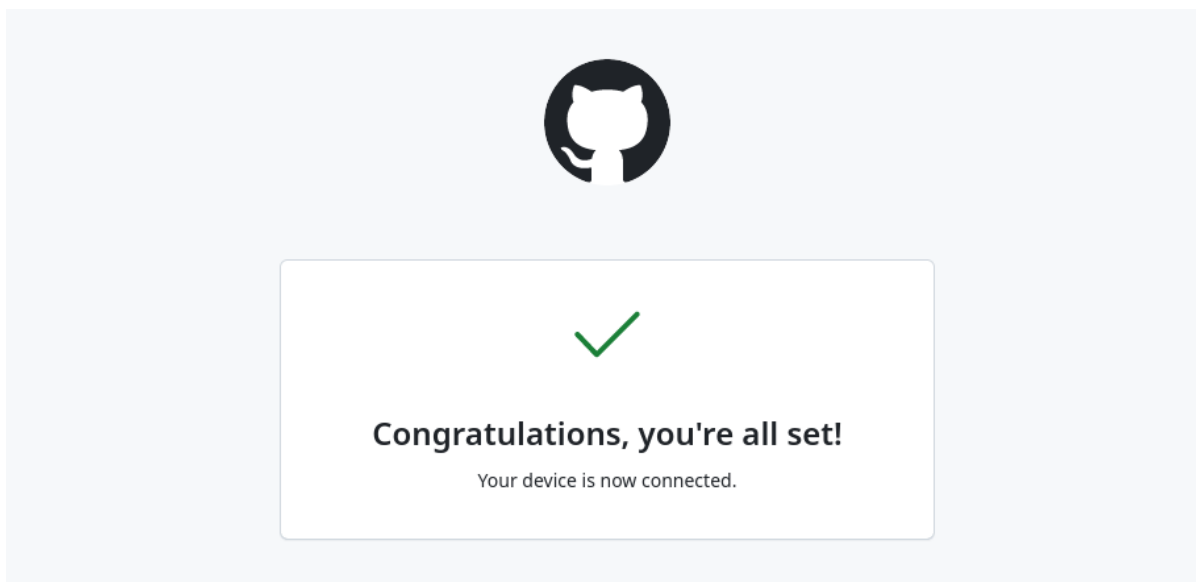


Рис. 4.10: Авторизация

Создаю репозиторий на GitHub.

```
mkdir -p ~/work/study/2023-2024/"Операционные системы"
```

```
cd ~/work/study/2023-2024/"Операционные системы"
```

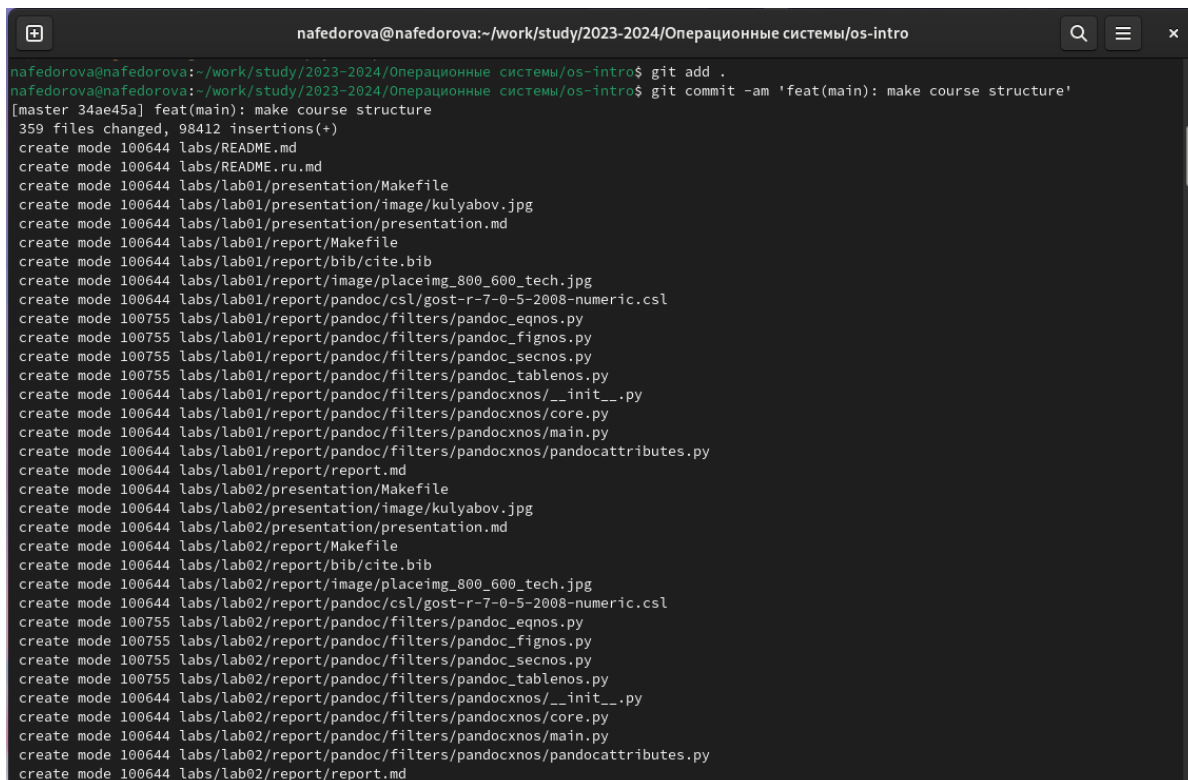
```
gh repo create study_2023-2024_os-intro --template=yamadharma/course-directory-  
student-template --public
```

```
git clone --recursive git@github.com:/study_2023-2024_os-intro.git os-intro (рис.  
4.11).
```

```
nafedorova@nafedorova:~/work/study/2023-2024/Операционны...
nafedorova@nafedorova:~$ mkdir -p ~/work/study/2023-2024/"Операционные системы"
nafedorova@nafedorova:~$ cd ~/work/study/2023-2024/"Операционные системы"
nafedorova@nafedorova:~/work/study/2023-2024/Операционные системы$ gh repo create s
tudy_2023-2024_os-intro --template=yamadharm/course-directory-student-template --p
ublic
✓ Created repository ChizuHira/study_2023-2024_os-intro on GitHub
https://github.com/ChizuHira/study_2023-2024_os-intro
nafedorova@nafedorova:~/work/study/2023-2024/Операционные системы$ git clone --recu
rsive git@github.com:ChizuHira/study_2023-2024_os-intro.git os-intro
Клонирование в «os-intro»...
The authenticity of host 'github.com (140.82.121.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvC0qU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
remote: Enumerating objects: 32, done.
remote: Counting objects: 100% (32/32), done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 32 (delta 1), reused 18 (delta 0), pack-reused 0
Получение объектов: 100% (32/32), 18.59 КиБ | 3.72 МиБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharm/academic-presentat
ion-markdown-template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharm/academic-laboratory-repo
rt-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/nafedorova/work/study/2023-2024/Операционные системы/os-intro
/template/presentation»...
remote: Enumerating objects: 95, done.
remote: Counting objects: 100% (95/95), done.
remote: Compressing objects: 100% (67/67), done.
remote: Total 95 (delta 34), reused 87 (delta 26), pack-reused 0
Получение объектов: 100% (95/95), 96.99 КиБ | 1.05 МиБ/с, готово.
Определение изменений: 100% (34/34), готово.
Клонирование в «/home/nafedorova/work/study/2023-2024/Операционные системы/os-intro
/template/report»...
remote: Enumerating objects: 126, done.
remote: Counting objects: 100% (126/126), done.
remote: Compressing objects: 100% (87/87), done.
remote: Total 126 (delta 52), reused 108 (delta 34), pack-reused 0
Получение объектов: 100% (126/126), 335.80 КиБ | 1.91 МиБ/с, готово.
Определение изменений: 100% (52/52), готово.
Submodule path 'template/presentation': checked out '40a1761813e197d00e8443ff1ca72c
60a304f24c'
Submodule path 'template/report': checked out '7c31ab8e5dfa8cdb2d67caeb8a19ef8028ce
d88e'
nafedorova@nafedorova:~/work/study/2023-2024/Операционные системы$
```

Рис. 4.11: Создание репозитория

Настраиваю каталог курса. Для этого сначала зайду в сам каталог, который скопировала до этого, с помощью команды **cd**, удалила лишние файлы **rm package.json**, создаю еще необходимые каталоги **echo os-intro > COURSE** после **make**, которые после отправила на сервер **git add .**, **git commit -am 'feat(main): make course structure'**, **git push** (рис. 4.12), (рис. 4.13).



```
nafedorova@nafedorova:~/work/study/2023-2024/Операционные системы/os-intro$ git add .
nafedorova@nafedorova:~/work/study/2023-2024/Операционные системы/os-intro$ git commit -am 'feat(main): make course structure'
[master 34ae45a] feat(main): make course structure
359 files changed, 98412 insertions(+)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
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/placeimg_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
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 labs/lab01/report/report.md
create mode 100644 labs/lab02/presentation/Makefile
create mode 100644 labs/lab02/presentation/image/kulyabov.jpg
create mode 100644 labs/lab02/presentation/presentation.md
create mode 100644 labs/lab02/report/Makefile
create mode 100644 labs/lab02/report/bib/cite.bib
create mode 100644 labs/lab02/report/image/placeimg_800_600_tech.jpg
create mode 100644 labs/lab02/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_fignos.py
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_secnos.py
create mode 100755 labs/lab02/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/core.py
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/main.py
create mode 100644 labs/lab02/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 labs/lab02/report/report.md
```

Рис. 4.12: Отправка на сервер

```
nafedorova@nafedorova:~/work/study/2023-2024/Операционные системы/os-intro
create mode 100644 project-personal/stage5/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage5/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 project-personal/stage5/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 project-personal/stage5/report/pandoc/filters/pandoc_fignos.py
create mode 100755 project-personal/stage5/report/pandoc/filters/pandoc_secnos.py
create mode 100755 project-personal/stage5/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 project-personal/stage5/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 project-personal/stage5/report/pandoc/filters/pandocxnos/core.py
create mode 100644 project-personal/stage5/report/pandoc/filters/pandocxnos/main.py
create mode 100644 project-personal/stage5/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 project-personal/stage5/report/report.md
create mode 100644 project-personal/stage6/presentation/Makefile
create mode 100644 project-personal/stage6/presentation/image/kulyabov.jpg
create mode 100644 project-personal/stage6/presentation/presentation.md
create mode 100644 project-personal/stage6/report/Makefile
create mode 100644 project-personal/stage6/report/bib/cite.bib
create mode 100644 project-personal/stage6/report/image/placeimg_800_600_tech.jpg
create mode 100644 project-personal/stage6/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_eqnos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_fignos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_secnos.py
create mode 100755 project-personal/stage6/report/pandoc/filters/pandoc_tablenos.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/core.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/main.py
create mode 100644 project-personal/stage6/report/pandoc/filters/pandocxnos/pandocattributes.py
create mode 100644 project-personal/stage6/report/report.md
nafedorova@nafedorova:~/work/study/2023-2024/Операционные системы/os-intro$ git push
Перечисление объектов: 39, готово.
Подсчет объектов: 100% (39/39), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (39/39), готово.
Запись объектов: 100% (38/38), 342.07 Киб | 2.80 Миб/с, готово.
Всего 38 (изменений 4), повторно использовано 1 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (4/4), completed with 1 local object.
To github.com:ChizuHira/study_2023-2024_os-intro.git
  8e51a27..34ae45a master -> master
nafedorova@nafedorova:~/work/study/2023-2024/Операционные системы/os-intro$
```

Рис. 4.13: Отправка на сервер

5 Контрольные вопросы

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

Система контроля версий (VCS) — это инструмент, использование которого позволяет отслеживать изменения в файловой системе, фиксировать историю изменений, а также возвращаться к предыдущим версиям файлов. Они предназначены для управления изменениями в проектах программного обеспечения и других файлов, позволяя команде разработчиков совместно работать над кодом, отслеживать изменения, управлять конфликтами и версиями, а также восстанавливать предыдущие состояния проекта.

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

Хранилище (репозиторий) — это централизованное место, где хранятся файлы и история изменений проекта. Оно содержит все версии файлов, метаданные и историю коммитов.

Commit (фиксация) — это действие по сохранению изменений в системе контроля версий. При коммите разработчик предоставляет описание внесенных изменений, и эти изменения фиксируются в репозитории.

История (history) — это список всех коммитов и изменений, связанных с проектом. История содержит информацию о том, кто, когда и какие изменения вносил, и позволяет отслеживать всю историю проекта.

Рабочая копия (working copy) — это локальная копия файлов из репозитория, с которой работает разработчик. Рабочая копия содержит текущую версию проекта, и разработчик вносит изменения в нее перед их фиксацией (коммитом) в репозиторий.

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

Централизованная система контроля версий (Centralized Version Control System, CVS) предполагает, что существует единый центральный репозиторий, в котором хранится вся история проекта. Разработчики работают с рабочими копиями файлов, которые забирают из центрального репозитория, вносят изменения и отправляют их обратно. Примеры централизованных VCS: CVS, Subversion (SVN), Perforce.

Децентрализованные системы контроля версий (Distributed Version Control System, DVCS) отличаются тем, что каждый разработчик имеет свою локальную копию репозитория, содержащую всю историю проекта. Это позволяет работать независимо от доступности центрального сервера и облегчает совместную работу над проектом. Примеры децентрализованных VCS: Git, Mercurial, Bazaar.

Основные отличия между централизованными и децентрализованными системами контроля версий заключаются в том, как управляется и хранится история версий проекта, а также в способе совместной работы разработчиков.

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

При единоличной работе с хранилищем VCS основными действиями будут:

- **Инициализация репозитория:** создание нового проекта или клонирование существующего репозитория из удаленного источника (например, GitHub).
- **Добавление файлов:** добавление новых файлов в репозиторий или изменение уже существующих файлов.
- **Фиксация изменений:** коммит изменений в репозиторий с указанием описания изменений.

- **Просмотр истории изменений:** просмотр и анализ всех предыдущих коммитов, внесенных в репозиторий.
- **Ветвление:** создание отдельных веток для разработки новых функций или исправлений багов.
- **Объединение изменений:** слияние веток и консолидация изменений в основной ветке разработки.
- **Удаление файлов:** удаление ненужных файлов из репозитория.
- **Удаленная работа:** отправка изменений на удаленный сервер и получение изменений из удаленного репозитория.

Все эти действия помогают эффективно управлять версиями кода и отслеживать изменения в проекте, даже при работе в одиночку.

5. Опишите порядок работы с общим хранилищем VCS.

- **Создание репозитория:** сначала необходимо создать репозиторий на сервере или в облаке, где будет храниться общее хранилище файлов.
- **Клонирование репозитория:** разработчики должны клонировать репозиторий себе на локальную машину, чтобы иметь доступ к файлам и иметь возможность вносить изменения.
- **Работа с файлами:** разработчики могут вносить изменения в файлы на локальной машине, создавать новые файлы, удалять или редактировать существующие.
- **Подготовка к коммиту:** перед сохранением изменений в репозиторий, необходимо подготовить их к коммиту, добавив их в “индекс” при помощи команды `git add`.

- **Коммит изменений:** после подготовки изменений, разработчики должны сделать коммит, сохраняя все внесенные изменения в историю репозитория при помощи команды `git commit`.
- **Пуш изменений:** после коммита, изменения могут быть отправлены в общее хранилище с помощью команды `git push`, что позволит другим разработчикам видеть и получать эти изменения.
- **Обновление локального репозитория:** разработчики могут получить последние изменения из общего хранилища с помощью команды `git pull`, чтобы обновить свою локальную версию репозитория.

Таким образом, порядок работы с общим хранилищем VCS заключается в клонировании, внесении изменений, коммите и отправке изменений в общее хранилище, а также в получении и обновлении локальной версии репозитория.

6. Каковы основные задачи, решаемые инструментальным средством `git`?

- Управление версиями файлов и их изменениями
- Совместная разработка проектов
- Отслеживание изменений и истории проекта
- Управление конфликтами при слиянии изменений
- Резервное копирование и восстановление данных

7. Назовите и дайте краткую характеристику командам `git`.

git init: инициализация нового репозитория

git add: добавление изменений в индекс

git commit: сохранение изменений в репозитории

git push: отправка изменений в удаленный репозиторий

git pull: получение изменений из удаленного репозитория

git branch: создание, удаление и просмотр веток

git merge: объединение изменений из другой ветки

git checkout: переключение между ветками

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

Локальный репозиторий: создание нового проекта с помощью `git init`, добавление новых файлов с помощью `git add`, сохранение изменений в репозитории с помощью `git commit`.

Удаленный репозиторий: отправка изменений из локального репозитория на удаленный с помощью `git push`, получение изменений из удаленного репозитория с помощью `git pull`.

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

Ветви (branches) в `git` используются для разработки новых функций, изоляции изменений, параллельной разработки, исправления ошибок и управления версиями проектов.

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

Для игнорирования некоторых файлов при `commit` в `git` используется файл `.gitignore`, в котором указываются шаблоны файлов или директорий, которые не должны попадать в репозиторий. Например, можно игнорировать временные файлы, файлы с настройками IDE, файлы с конфиденциальной информацией и т.д. # Выводы

В данной лабораторной работе я изучила методы применения средств контроля версий. Освоила умения по работе с `git`.