

# **Лабораторная работа №2**

**Первоначальная настройка git**

Тойчубекова Асель Нурлановна

# 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>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>10</b>
4.1	Установка программного обеспечения . . . . .	10
4.2	Базовая настройка git. . . . .	10
4.3	Создание ключей ssh . . . . .	11
4.4	Создание ключа pgr . . . . .	12
4.5	Настройка github . . . . .	13
4.6	Добавление PGP ключа в GitHub . . . . .	14
4.7	Настройка автоматических подписей коммитов git . . . . .	16
4.8	Настройка gh . . . . .	16
4.9	Создание репозитория курса на основе шаблона . . . . .	17
4.10	Настройка каталога курса . . . . .	18
4.11	Ответы на контрольные вопросы . . . . .	20
<b>5</b>	<b>Выводы</b>	<b>25</b>
	<b>Список литературы</b>	<b>26</b>

# Список иллюстраций

3.1	Основные команды git. . . . .	9
4.1	Установка git . . . . .	10
4.2	Установка gh . . . . .	10
4.3	Установка имени,email пользователя и utf-8 . . . . .	11
4.4	Задание начальной ветки и настройка параметров autocrlf, safecrlf . . . . .	11
4.5	Создание ssh ключей. . . . .	12
4.6	Создание pgr ключа . . . . .	13
4.7	Создание pgr ключа . . . . .	13
4.8	Аккаунт в github . . . . .	14
4.9	Список ключей . . . . .	14
4.10	Копирование PGP ключ . . . . .	15
4.11	Раздел New GPG key . . . . .	15
4.12	Сгенерированный ключ . . . . .	16
4.13	Настройка автоматических подписей коммитов git . . . . .	16
4.14	Авторизация через браузер . . . . .	17
4.15	Авторизация через браузер . . . . .	17
4.16	Создание каталога . . . . .	17
4.17	Создание репозитория . . . . .	18
4.18	Клонирование репозитория . . . . .	18
4.19	Удаление лишнего файла . . . . .	18
4.20	Создание каталогов . . . . .	19
4.21	Отправление файла на сервер . . . . .	19
4.22	Репозиторий в github . . . . .	20

## Список таблиц

# 1 Цель работы

Целью лабораторной работы №2 является изучение идеологии и применение средств контроля версий, а также освоение умений по работе с git.

## 2 Задание

- Создать базовую конфигурацию для работы с git.
- Создать ключ SSH.
- Создать ключ PGP.
- Настроить подписи git.
- Зарегистрироваться на Github.
- 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.

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. 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 фикси-

ровать, 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.

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

Ниже на фото представлены основные команды `git`. (рис. 3.1).



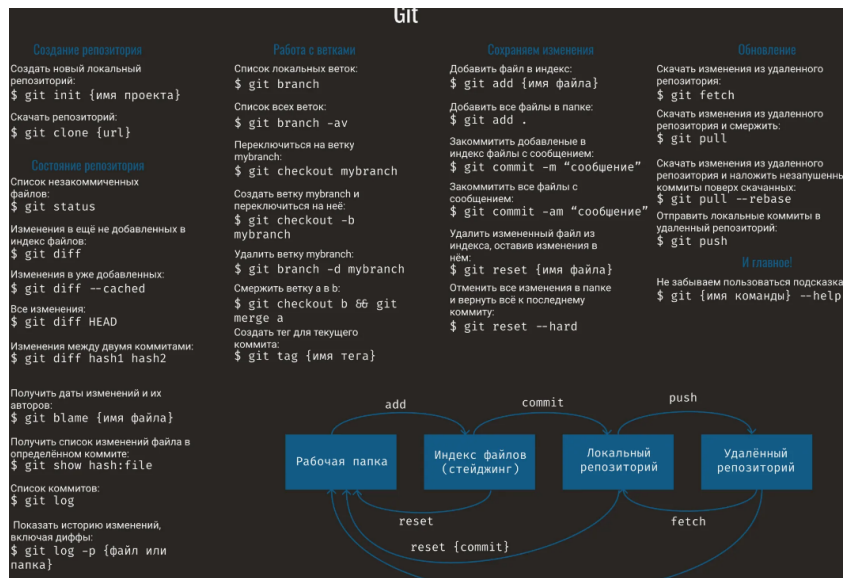


Рис. 3.1: Основные команды git.

## 4 Выполнение лабораторной работы

### 4.1 Установка программного обеспечения

Сперва захожу в терминал и устанавливаю git,используя команду ‘dnf install git’. (рис. 4.1)

```
[antoyjchubekova@antoyjchubekova ~]$ sudo dnf install git
[sudo] пароль для antoyjchubekova:
Fedora 39 - x86_64 - Updates
Fedora 39 - x86_64 - Updates
Последняя проверка окончания срока действия метаданных: 0:00:00 назад, Вс 25 фев 2024 16:27:52.
Пакет git-2.43.2-1.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
```

Рис. 4.1: Установка git

Устанавливаю gh, используя команду ‘dnf install gh’. (рис. 4.2)

```
[antoyjchubekova@antoyjchubekova ~]$ sudo dnf install gh
Последняя проверка окончания срока действия метаданных: 0:00:56 назад, Вс 25 фев 2024 16:27:52.
Пакет gh-2.43.1-1.fc39.x86_64 уже установлен.
Зависимости разрешены.
Нет действий для выполнения.
Выполнено!
```

Рис. 4.2: Установка gh

### 4.2 Базовая настройка git.

Пользуясь командой ‘git config –global user.name “Asel Toychubekova” ’ задаю свое имя, а командой ‘git config –global user.email “aseltoychubekova714@gmail.com” ’ задаю свой email для репозитория, а также настраиваю utf-8 в выводе сообщений git командой ‘git config –global core.quotePath false’. (рис. 4.3)

```
[antoyjchubekova@antoyjchubekova ~]$ git config --global user.name "AseI Toichubekova"
[antoyjchubekova@antoyjchubekova ~]$ git config --global user.email 'aseltoychubekova714@gmail.com'
[antoyjchubekova@antoyjchubekova ~]$ git config --global core.quotepath false
```

Рис. 4.3: Установка имени, email пользователя и utf-8

Задаю имя начальной ветки, название-master, командой 'git config --global init.defaultBranch master', затем настраиваю параметры autocrlf( git config --global core.autocrlf input) и safecrlf(git config --global core.safecrlf warn). (рис. 4.4)

```
[antoyjchubekova@antoyjchubekova ~]$ git config --global init.defaultBranch master
[antoyjchubekova@antoyjchubekova ~]$ git config --global core.autocrlf input
[antoyjchubekova@antoyjchubekova ~]$ git config --global core.safecrlf warn
```

Рис. 4.4: Задание начальной ветки и настройка параметров autocrlf, safecrlf

### 4.3 Создание ключей ssh

Создаю ключ по алгоритму rsa с ключём размером 4096 бит, после чего создаю ключ по алгоритму ed25519, используя команду 'ssh keygen -t'. (рис. 4.5)

```

[antoyjchubekova@antoyjchubekova ~]$ ssh-keygen -t rsa -b 4096
ssh: Could not resolve hostname keygen: Name or service not known
[antoyjchubekova@antoyjchubekova ~]$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/antoyjchubekova/.ssh/id_rsa):
Created directory '/home/antoyjchubekova/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/antoyjchubekova/.ssh/id_rsa
Your public key has been saved in /home/antoyjchubekova/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:pxujy0TiaTqPKm4txso0p0RWN3UyPqM4xSvYpxu5nUE antoyjchubekova@antoyjchubekova
The key's randomart image is:
+---[RSA 4096]-----+
|
|  + o
|  + +
|  = +
|  . =
| o ..o.ES .
|o .++= o
| .+.o+.o+
|=== o+.o+
|0=+o.+++.
+---[SHA256]-----+
[antoyjchubekova@antoyjchubekova ~]$ ssh-keygen -t ed25519
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/antoyjchubekova/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/antoyjchubekova/.ssh/id_ed25519
Your public key has been saved in /home/antoyjchubekova/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:6HYDN5VpxfDjUihY20dVtYo81FpeoNU45g/2bf9NZN antoyjchubekova@antoyjchubekova
The key's randomart image is:
+--[ED25519 256]--+
|
| .o+o+oB|
| BB0o++|
| . B0+o+o|
| o = +.++o|
| o S . .+.o|
| . o . . . =|
| o o . Eo|

```

Рис. 4.5: Создание ssh ключей.

## 4.4 Создание ключа gpg

Генерирую ключ, командой 'gpg --full-generate-key', выбирая подходящие из предложенных опций. (рис. 4.6 и рис. 4.7)

```
[antojychubekova@antojychubekova ~]$ 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/antojychubekova/.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)
Срок действия ключа не ограничен
Все верно? (y/N) y

GnuPG должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Asel Toichubekova
Адрес электронной почты: aseltoychubekova714@gmail.com
Примечание: asel
Вы выбрали следующий идентификатор пользователя:
  "Asel Toichubekova (asel) <aseltoychubekova714@gmail.com>"

Сменить (N)Имя, (C)Примечание, (E)Адрес; (O)Принять/(Q)Выход? o
```

Рис. 4.6: Создание pgr ключа

```
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/antojychubekova/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/home/antojychubekova/.gnupg/openpgp-revocs.d'
gpg: сертификат отзыва записан в '/home/antojychubekova/.gnupg/openpgp-revocs.d/0A5B8496C6B2D2895CC0B4D72E36AB2269233CC8.rev'
открытый и секретный ключи созданы и подписаны.

pub   rsa4096 2024-02-25 [SC]
      8A5B8496C6B2D2895CC0B4D72E36AB2269233CC8
uid           Asel Toichubekova (asel) <aseltoychubekova714@gmail.com>
sub   rsa4096 2024-02-25 [E]
```

Рис. 4.7: Создание pgr ключа

## 4.5 Настройка github

У меня уже был настроен github, я создала учетную запись заполнила основные данные на прошлом семестре. (рис. 4.8)

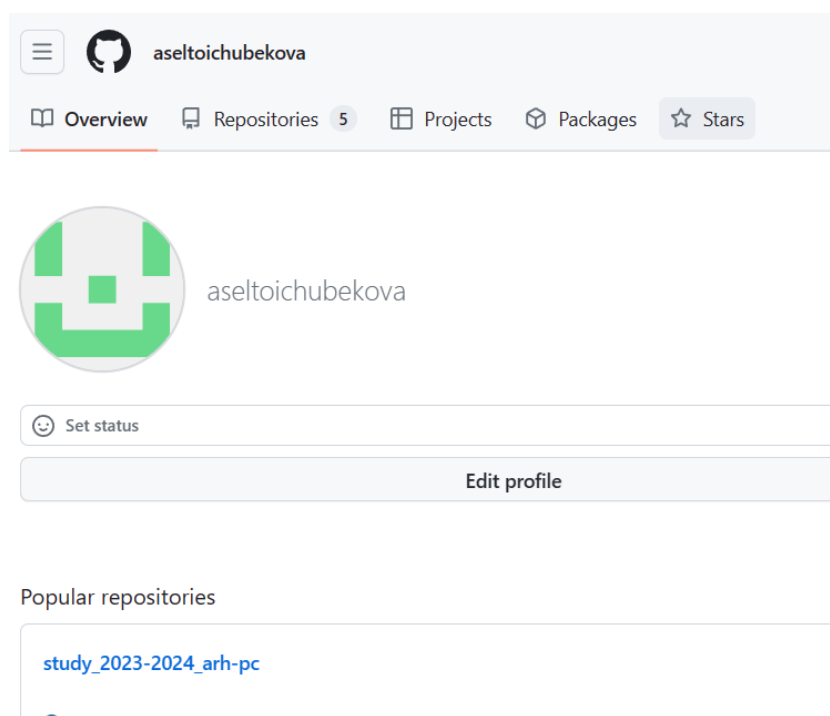


Рис. 4.8: Аккаунт в github

## 4.6 Добавление PGP ключа в GitHub

Вывожу список ключей, командой 'gpg --list-secret-keys --keyid-format LONG'.  
(рис. 4.9)

```
(antoyjchubekova@antoyjchubekova ~)$ gpg --list-secret-key --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, 0f, 1u
[keyboard]
-----
sec rsa4096/2E36A82269233CC8 2024-02-25 [5C]
      0A5B0496C6B2D2895CC0B4D72E36A82269233CC8
uid [ абсолютно ] Asel Toichubekova (asel) <aselttoichubekova714@gmail.com>
ssb rsa4096/4EA7573ECDE05D8E 2024-02-25 [E]
```

Рис. 4.9: Список ключей

Копирую сгенерированный PGP ключ в буфер обмена, командой 'gpg --armor --export ключ | xclip -sel clip', перед этим установив команду xclip. (рис. 4.10)

```
[antoychubekova@antoychubekova ~]$ sudo dnf install kclip
Последняя проверка окончания срока действия метаданных: 0:09:09 назад, Вс 25 фев 2024 17:01:20.
Нет соответствия аргументу: kclip
Ошибка: Скомпаний не найдено: kclip
[antoychubekova@antoychubekova ~]$ sudo dnf install kclip
Последняя проверка окончания срока действия метаданных: 0:09:49 назад, Вс 25 фев 2024 17:01:20.
Зависимости разрешены.

=====
Пакет      Архитектура  Версия      Резепозиторий  Размер
=====
Установка:
kclip      x86_64       0.13-20.git11c6a61.fc39  Fedora        37 k
=====
Результат транзакции
Установка 1 пакет

Объем загрузки: 37 k
Объем изменений: 62 k
Продолжить? [A/N]: d
Загрузка пакетов:
kclip-0.13-20.git11c6a61.fc39.x86_64.rpm                222 kB/s | 37 kB  00:00
Общий размер
kclip-0.13-20.git11c6a61.fc39.x86_64.rpm                45 kB/s | 37 kB  00:00
Проверка транзакции
Проверка транзакции успешно завершена.
Идет проверка транзакции
Тест транзакции пройден успешно.
Выполнение транзакции
Подготовка:
Установка: kclip-0.13-20.git11c6a61.fc39.x86_64          1/1
Запуск скрипта: kclip-0.13-20.git11c6a61.fc39.x86_64      1/1
Проверка: kclip-0.13-20.git11c6a61.fc39.x86_64           1/1
Установлено:
kclip-0.13-20.git11c6a61.fc39.x86_64
Выполнено!
[antoychubekova@antoychubekova ~]$ gpg --armor --export 2E36AB2269233CC8 | kclip --set c1ip
```

Рис. 4.10: Копирование PGP ключ

Перехожу в настройки GitHub, нажимаю на кнопку New GPG key и вставляю полученный ключ в поле ввода. (рис. 4.11) , далее получаю сгенерированный ключ (рис. 4.12)

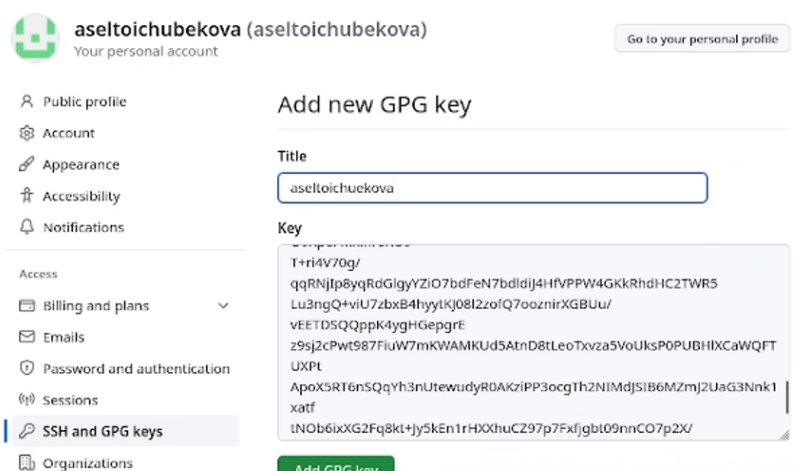


Рис. 4.11: Раздел New GPG key

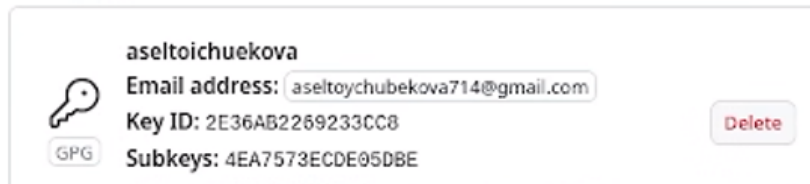


Рис. 4.12: Сгенерированный ключ

## 4.7 Настройка автоматических подписей коммитов git

Используя введённый email, укажем Git применять его при подписи коммитов (`git config --global user.signingkey email`) (рис. 4.13)

```
[antoychubekova@antoychubekova ~]$ git config --global user.signingkey aselttoychubekova714@gmail.com
[antoychubekova@antoychubekova ~]$ git config --global commit.gpgsign true
[antoychubekova@antoychubekova ~]$ git config --global gpg.program $(which gpg2)
```

Рис. 4.13: Настройка автоматических подписей коммитов git

## 4.8 Настройка gh

Используя команду '`gh auth login`', авторизуюсь через браузер (рис. 4.14 и рис. 4.15)



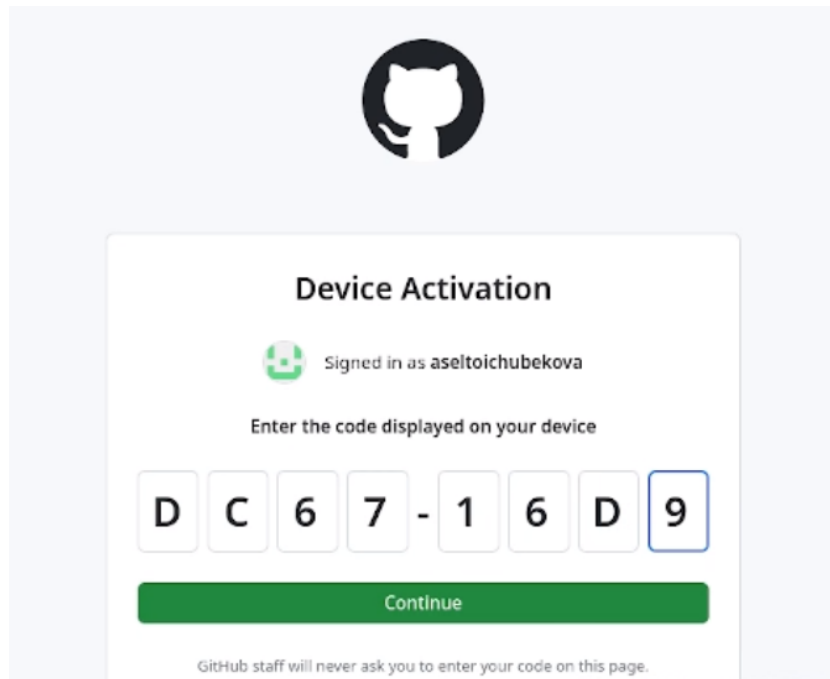


Рис. 4.14: Авторизация через браузер

```
[antoyjchubekova@antoyjchubekova ~]$ gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations on this host? SSH
? Upload your SSH public key to your GitHub account? /home/antoyjchubekova/.ssh/id_rsa.pub
? Title for your SSH key: aseltoichubekova
? How would you like to authenticate GitHub CLI? Login with a web browser
First copy your one-time code: DC67-16D9
Press Enter to open github.com in your browser...
```

Рис. 4.15: Авторизация через браузер

## 4.9 Создание репозитория курса на основе шаблона

Создаю каталог, в котором мы будем дальше работать, перехожу в него, используя команды `mkdir` и `cd` (рис. 4.16)

```
[antoyjchubekova@antoyjchubekova ~]$ mkdir -p ~/work/study/2023-2024/"Операционные системы"
[antoyjchubekova@antoyjchubekova ~]$ cd ~/work/study/2023-2024/"Операционные системы"
[antoyjchubekova@antoyjchubekova Операционные системы]$
```

Рис. 4.16: Создание каталога

Далее создаю репозиторий на основе шаблона, с помощью команды `'gh repo`

create study\_2022-2023\_os-intro --template=yamadharm/course-directory-student-template --public' (рис. 4.17)

```
[antoychubekova@antoychubekova ~]$ gh repo create study_2022-2023_os-intro --template=yamadharm/course-directory-student-template --public
Created repository aseltoichubekova/study_2022-2023_os-intro on GitHub
https://github.com/aseltoichubekova/study_2022-2023_os-intro
```

Рис. 4.17: Создание репозитория

После кланирую репозиторий,командой 'git clone --recursive git@github.com:aseltoichubekova/study\_2023-os-intro.git os-intro' (рис. 4.18)

```
[antoychubekova@antoychubekova Операционные системы]$ git clone --recursive git@github.com:aseltoichubekova/study_2023-2024_os-intro.git os-intro
Клонирование в «os-intro»...
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), 18.60 КиБ | 4.65 МБ/с, готово.
Определение изменений: 100% (1/1), готово.
Подмодуль «template/presentation» (https://github.com/yamadharm/academic-presentation.markdown.template.git) зарегистрирован по пути «template/presentation»
Подмодуль «template/report» (https://github.com/yamadharm/academic-laboratory-report-template.git) зарегистрирован по пути «template/report»
Клонирование в «/home/antoychubekova/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 | 902.00 КиБ/с, готово.
Определение изменений: 100% (34/34), готово.
Клонирование в «/home/antoychubekova/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 | 2.04 МБ/с, готово.
Определение изменений: 100% (52/52), готово.
Submodule path 'template/presentation': checked out '40a1761813e197d00e84437f1ca72c60a304f24c'
Submodule path 'template/report': checked out '7c31ab8e5d8c8db2d67cae8a19ef8026ced88e'
```

Рис. 4.18: Клонирование репозитория

## 4.10 Настройка каталога курса

Перехожу в каталог курса -> cd ~/work/study/2022-2023/“Операционные системы”/os-intro и удаляю лишний файл rm package.json (рис 4.19)

```
[antoychubekova@antoychubekova Операционные системы]$ cd os-intro
[antoychubekova@antoychubekova os-intro]$ rm package.json
[antoychubekova@antoychubekova os-intro]$ ls
CHANGELOG.md  config  COURSE  LICENSE  Makefile  README.en.md  README.git-flow.md  README.md  template
```

Рис. 4.19: Удаление лишнего файла

Создаю необходимые каталоги, используя команды : echo os-intro > COURSE, затем make (рис 4.20)

```
[antoyjchubekova@antoyjchubekova os-intro]$ echo os-intro > COURSE
[antoyjchubekova@antoyjchubekova os-intro]$ make
Usage:
  make <target>

Targets:
  list              List of courses
  prepare           Generate directories structure
  submodule         Update submodules
[antoyjchubekova@antoyjchubekova os-intro]$ make prepare
```

Рис. 4.20: Создание каталогов

Отправляю файлы на сервер:(рис 4.21) - git add .

- git commit -am 'feat(main): make course structure'

- git push.

```
[antoyjchubekova@antoyjchubekova os-intro]$ git add .
[antoyjchubekova@antoyjchubekova os-intro]$ git commit -am 'feat(main): make course structure'
[master 167046b] feat(main): make course structure
361 files changed, 98413 insertions(+), 14 deletions(-)
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/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
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/placing_800_600_tech.jpg
create mode 100644 labs/lab02/report/pandoc/csl/gost-r-7-0-5-2008-numeric.csl
```

Рис. 4.21: Отправление файла на сервер

Зайдем в github и видим репозиторий созданный по шаблону (рис 4.22)

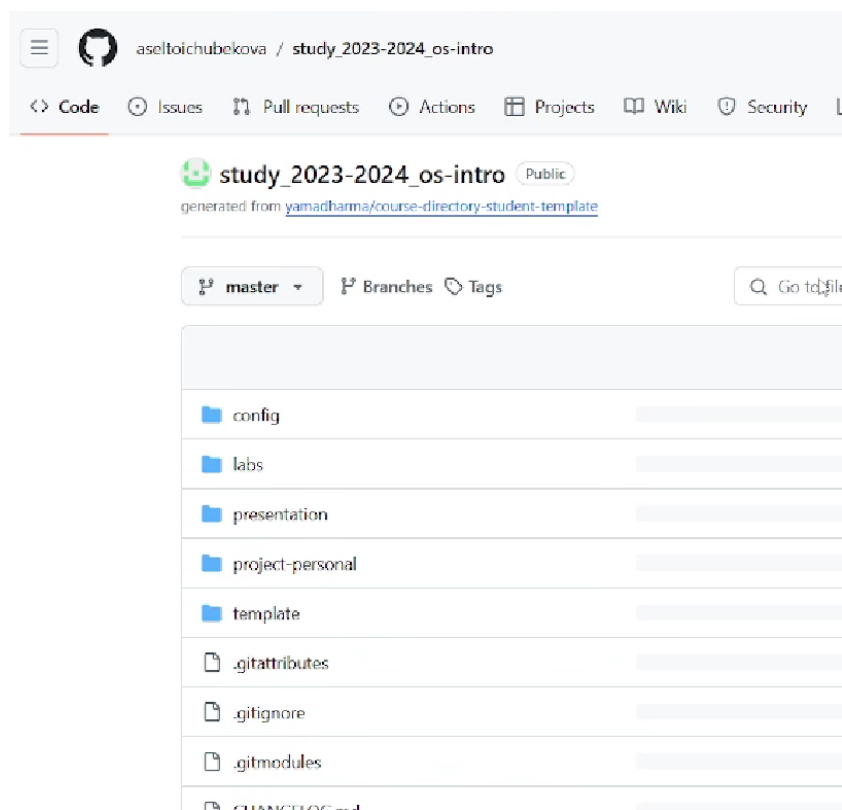


Рис. 4.22: Репозиторий в github

## 4.11 Ответы на контрольные вопросы

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

Системы контроля(VCS)- программное обеспечение для облегчения работы с изменяющейся информацией, например, когда над проектом работают несколько человек. Они позволяют хранить несколько версий информации, также можно посмотреть ранние версии этой информации-проекта. Это программное обеспечение позволяет просматривать кем и когда были внесены изменения в тот или иной проект. Его применяют для хранения полной истории именений, сохранения причин изменений, для удобства работы над проектом нескольких людей.

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

Хранилище- репозиторий, где сохраняются все версии проекта, в нем хранятся все документы, история их изменения и прочие служебные информации.

Commit-Отлаживает все изменения, сохраняет разницу версий, их изменения.

История-хранит все изменения в проекте и позволяет при необходимости использовать нужные данные из прошлых версий.

Рабочая копия-копия проекта, основанная на версии из хранилища, последней версии.

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

Централизованные VCS- одно основное хранилище всего проекта. Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет затем добавляет изменения обратно в хранилище.

Децентрализованные VCS- в этих VCS у каждого пользователя свой вариант репозитория, есть возможность добавлять и забирать изменения из любого репозитория.

В отличие от классических в децентрализованных системах контроля версий центральный репозиторий не является обязательным.

Централизованные VCS -CVS,TFS,AccuRev.

Децентрализованные VCS-Git,Bazaar.

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

Для начала для всех создается и подключается центральный репозиторий, затем по мере изменений проекта эти изменения отправляются на сервер.

5. Опишите порядок работы с общим хранилищем VCS.

Перед началом работы все участники проекта получают версию проекта в хранилище, после изменений пользователь размещает уже новую версию в хранилище. При этом предыдущие версии тоже сохраняются.

6. Каковы основные задачи, решаемые инструментальным средством git?

Основными задачами инструментального средства git является: хранение информации о всех изменениях, обеспечения удобства командной работы над проектом.

7. Назовите и дайте краткую характеристику командам git.

- Создания основного дерева репозитория-git init.
- Получение обновлений -git pull.
- Просмотр списка измененных файлов-git status.
- Отправка изменений локального дерева- git push.
- Просмотр текущих изменений-git diff.
- Сохранение изменений-git add .
- Добавить конкретные изменения -git add имя файла
- Удаление файл или каталог из индекса репозитория-git rm имя файла

- Сохранение добавленных изменений и все измененные файлы -git commit -am 'описание коммита'
- Сохранить добавленные изменения с внесением комментария через встроенный редактор- git commit
- Создание новой ветки базиркующаяся на текущей-git checkout-b имя ветки
- Переключение на новую ветку-git checkout имя ветки
- Отправка изменений конкретной ветки в центральный репозиторий- git push origin имя ветки
- Удаление локальной уже слитой с основным деревом ветки-git branch d- имя ветки
- Принудительное удаление ветки-git branch -D имя ветки
- Удаление ветки с центрального репозитория - go=it push origin: имя ветки.
- Слияние ветки с текущим деревом-git merge -no-ff имя веткию

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

git push-all отправляет из локального репозитория все сохраненные изменения в центральный репозиторий,предворительно создав локальный ррепозиторий и сделав предварительную конфигурацию.

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

Ветвление- один из параллельных участков в одном хранилище,исходящие из одной версии, обычно есть главная ветка, также возможно их слияние. Используется для разработки новых функций.

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

Когда идет работа над проектом могут создаваться файлы которые не нуужно добавлять в репозиторий. К ним относятся временные файлы. Можно прописать шаблоны игнарируемые при добавлении в репозиторий типов файлов в файл.gitignore с помощью сервисов.



## 5 Выводы

В ходе выполнения лабораторной работы №2 я изучила идеологию и примечание средств контроля версии, а также освоила умения для работы с git

## Список литературы

- <https://esystem.rudn.ru/mod/page/view.php?id=1098933#org2151722>.