

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ

### ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

дисциплина:     Архитектура компьютера

Студент: Гафоров Нурмухаммад

Группа: НПИбд-03-23

МОСКВА

20\_\_\_\_ г.

# Содержание

1.Цел работы .....	5
2.Задание.....	6
3.Теоритическое введение.....	7
4.Выполнение лабораторный работы.....	9
4.1 Базовая настройка Git.....	10
4.2 Создание SSH-ключа.....	11
4.3 Создание рабочего пространство и репозитория курса на основе шаблона.....	13
4.4 Создание репозитория курса на основе шаблона.....	14
4.5 Настройка каталога курса.....	18
4.6 Выполнение заданий для самостоятельной работы.....	20
5.Выводы.....	27
6.Список литературы.....	28

## Список иллюстраций

4.1 Аккаунт GitHub .....	9
4.2 Предварительная конфигурация git .....	10
4.3 Настройка кодировки .....	10
4.4 Создание имени для начальной ветки .....	10
4.5 Параметр autocrlf .....	10
4.6 Параметр safecrlf .....	10
4.7 Генерация SSH-ключа .....	11
4.8 Копирование содержимого файла .....	11
4.9 Окно SSH and GPG keys .....	12
4.10 Добавление ключа .....	13
4.11 Создание рабочего пространства .....	13
4.12 Страница шаблона для репозитория .....	14
4.13 Окно создания репозитория .....	15
4.14 Созданный репозиторий .....	16
4.15 Перемещение между директориями .....	16
4.16 Клонирование репозитория .....	17
4.17 Окно с ссылкой для копирования репозитория .....	18
4.18 Перемещение между директориями .....	18
4.19 Удаление файлов .....	18
4.20 Создание каталогов .....	19
4.21 Добавление и сохранение изменений на сервере .....	19
4.22 Выгрузка изменений на сервер .....	20
4.23 Страница репозитория .....	20
4.24 Создание файла .....	20
4.25 Меню приложений .....	21
4.26 Работа с отчетом в текстовом процессоре.....	21
4.27 Перемещение между директориями .....	22

4.28 Проверка местонахождения файлов .....	22
4.29 Копирование файла .....	22
4.30 Перемещение между директориями .....	22
4.31 Копирование файла .....	23
4.32 Добавление файла на сервер .....	23
4.33 Перемещение между директориями .....	23
4.34 Добавление файла на сервер .....	23
4.35 Подкаталоги и файлы в репозитории.....	24
4.36 Отправка в центральный репозиторий сохраненных изменений .....	24
4.37 Страница каталога в репозитории .....	24
4.38 Страница последних изменений в репозитории .....	25
4.39 Каталог lab01/report .....	25
4.40 Каталог lab02/report .....	25
4.41 Каталог lab03/report.....	26

## **1 Цель работы**

Целью данной работы является изучить идеологию и применение средств контроля версий, а также приобрести практические навыки по работе с системой `git`.

## **2 Задание**

1. Настройка GitHub.
2. Базовая настройка Git.
3. Создание SSH-ключа.
4. Создание рабочего пространства и репозитория курса на основе шаблона.
5. Создание репозитория курса на основе шаблона.
6. Настройка каталога курса.
7. Выполнение заданий для самостоятельной работы.

### 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 поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить изменения, сделанные разными участниками, 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. Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений

каждой ветви. Обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным. 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. 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. Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория (при этом в локальное дерево до начала этой процедуры не должно было вноситься изменений). Затем можно вносить изменения в локальном дереве и/или ветке. После завершения внесения какого-то изменения в файлы и/или каталоги проекта необходимо разместить их в центральном репозитории.



## 4 Выполнение лабораторной работы

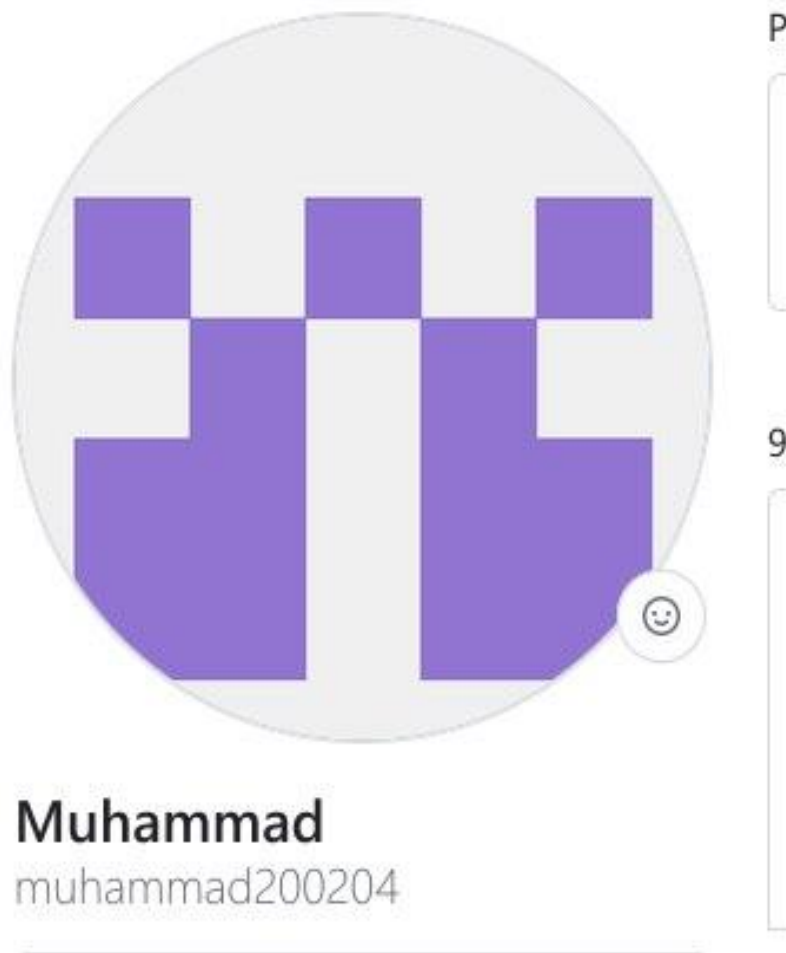


Рис.4.1 Аккаунт GitHub

### 4.1 Базовая настройка Git

Открываю виртуальную машину, затем открываю терминал и делаю предварительную конфигурацию git. Ввожу команду `git config --global user.name ""`, указывая свое имя и команду `git config --global user.email "work@mail"`, указывая в ней электронную почту владельца, то есть мою (рис. 4.2)

```
nvgaforov@dk3n35 ~ $ git config --global user.name "<Muhammad200204>"
nvgaforov@dk3n35 ~ $ git config --global user.email "<nurmuhammad.gaforov@bk.ru>"
```

Рис. 4.2: Предварительная конфигурация git

Настраиваю utf-8 в выводе сообщений git для корректного отображения символов (рис. 4.3).

```
nvgaforov@dk8n60 ~ $ git config --global core.quotepath false
```

Рис. 4.3: Настройка кодировки

Задаю имя «master» для начальной ветки (рис. 4.4).

```
nvgaforov@dk8n60 ~ $ git config --global init.defaultBranch master
```

Рис. 4.4: Создание имени для начальной ветки

Задаю параметр autocrlf со значением input, так как я работаю в системе Linux, чтобы конвертировать CRLF в LF только при коммитах (рис. 4.5). CR и LF – это символы, которые можно использовать для обозначения разрыва строки в текстовых файлах.

```
nvgaforov@dk8n60 ~ $ git config --global core.autocrlf input
```

Рис. 4.5: Параметр autocrlf

Задаю параметр safecrlf со значением warn, так Git будет проверять преобразование на обратимость (рис. 4.6). При значении warn Git только выведет предупреждение, но будет принимать необратимые конвертации.

```
nvgaforov@dk8n60 ~ $ git config --global core.safecrlf warn
```

## 4.2 Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый). Для этого ввожу команду `ssh-keygen -C "Имя Фамилия, work@email"`, указывая имя владельца и электронную почту владельца (рис. 4.7). Ключ автоматически сохранится в каталоге `~/.ssh/`.

```
nvgaforov@dk8n60 ~ $ ssh-keygen -C "Нурмухаммад Гафоров <nurmuhammad.gaforov@bk.ru>"
Generating public/private rsa key pair.
Enter file in which to save the key (/afs/.dk.sci.pfu.edu.ru/home/n/v/nvgaforov/.ssh/id_rsa):
/afs/.dk.sci.pfu.edu.ru/home/n/v/nvgaforov/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /afs/.dk.sci.pfu.edu.ru/home/n/v/nvgaforov/.ssh/id_rsa
Your public key has been saved in /afs/.dk.sci.pfu.edu.ru/home/n/v/nvgaforov/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:1aINp2DBNx8v4NufmSut1N5zVZ1cK2jwi+PXdz0H9cE Нурмухаммад Гафоров <nurmuhammad.gaforov@bk.ru>
The key's randomart image is:
+---[RSA 3072]-----+
|  ..                |
| ..+ .. .          |
| oo.+oo= . . =     |
| . . *oo.+ . +o    |
| ooS.o . o .       |
| . .+ . Eo         |
|   ooo+. +o        |
| ..o*o..o++       |
| .o+..o. =        |
+----[SHA256]-----+
nvgaforov@dk8n60 ~ $
```

Рис. 4.7: Генерация SSH-ключа

Копирую открытый ключ из директории, в которой он был сохранен, с помощью утилиты `xclip` (рис. 4.8).

```
nvgaforov@dk3n35 ~ $ cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

Рис. 4.8: Копирование содержимого файла

Открываю браузер, захожу на сайт GitHub. Открываю свой профиль и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key» (рис. 4.9).

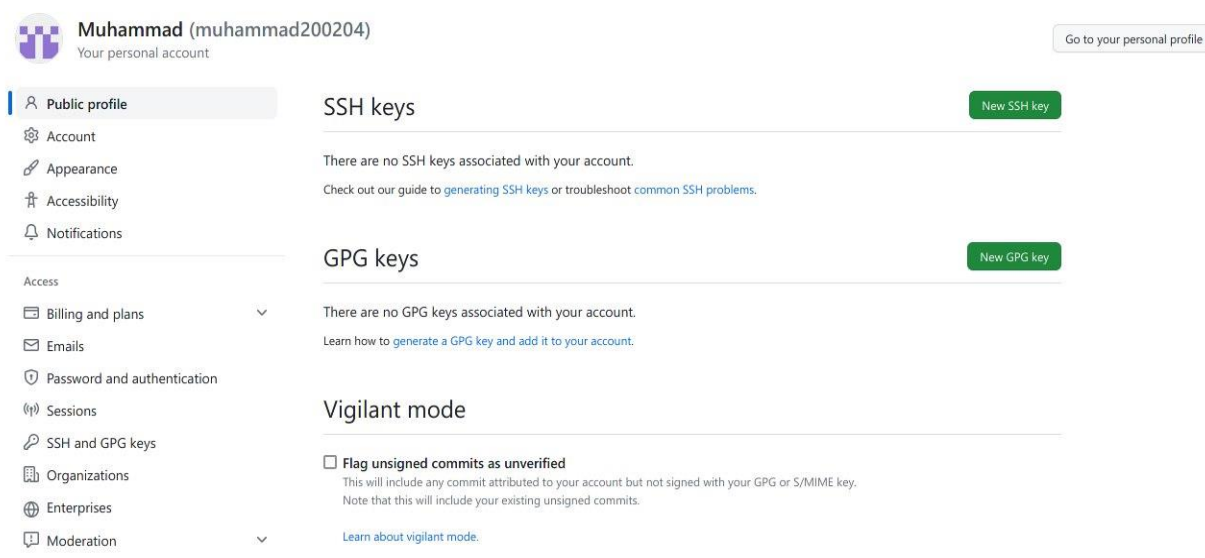


Рис. 4.9: Окно SSH and GPG keys

Вставляю скопированный ключ в поле «Key». В поле Title указываю имя для ключа.

Нажимаю «Add SSH-key», чтобы завершить добавление ключа (рис. 4.10)

## Add new SSH Key

Title

Key type

Authentication Key

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQGDQSKJnG7zUwBDsG6t5T+dB4qsT3PtgWylQvPaPcTXY4SMM7Lw81hgDf6ulDliOd
LjMKzfD8jboDniKwRz2Wlt4m9aE5zimUfxr9lLqFl0mW7LL9VOoig7Zhv6AspUvrKkgfLCBnlisNzFz+Wr88T/5hCv5fT69zK7zoix
Yo4xpl0DJ2oeGzzFuuak1rvWXcBEluEZdlLgBsQJqF6DdTUOf0ozo/gjUBdLmnDUlt9DIFlon
/46A8Jtrt7JlgzhtJauelW7UIRaPzDliLZxQRu7JkjaUBLSM9y1fqQP7AFM6vnqa5+p7r69fPv85ZKeiKaODBqcFTkZlwhkC4C5ye
poE7XjOca7Ao2/LSgrexhscq1qH5bAMI7TxxEqfX7DzL+XcdF4nIU8N7i7G4dS66vDBqS172CFoGS05uSVkbfYrLZwsl8Dx+
/TFDDRfH8+hUITGBkwQDhHHT+YHl1maC1yvjqdf6C2WjLIR5UdlTfk+DHbd7UdfTwskmrsnpXe7FE= muhammad200204
<nurmuhammad.gaforov@bk.ru>
```

Add SSH key

Рис. 4.10: Добавление ключа

## 4.3 Создание рабочего пространства и репозитория курса на основе шаблона

Закрываю браузер, открываю терминал. Создаю директорию, рабочее пространство, с помощью утилиты `mkdir`, благодаря ключу `-p` создаю все директории после домашней `~/work/study/2022-2023/` “Архитектура компьютера” рекурсивно. Далее проверяю с помощью `ls`, действительно ли были созданы необходимые мне каталоги (рис. 4.11).

```
nvgafarov@dk3n35 ~$ mkdir -p work/study/2023-2024/"Архитектура компьютера"
nvgafarov@dk3n35 ~$ ls
work Видео Документы Изображения Музыка Рабочий стол Шаблоны
nvgafarov@dk3n35 ~$
```

Рис. 4.11: Создание рабочего пространства

## 4.4 Создание репозитория курса на основе шаблона

В браузере перехожу на страницу репозитория с шаблоном курса по адресу

<https://github.com/yamadharm/course-directory-student-template>. Далее выбираю «Use this template»,

чтобы использовать этот шаблон для своего репозитория (рис. 4.12).


**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere?  
[Import a repository.](#)

---

*Required fields are marked with an asterisk (\*).*

**Repository template**


 yamadharm/course-directory-student-template ▾

Start your repository with a template repository's contents.

☐ **Include all branches**  
Copy all branches from yamadharm/course-directory-student-template and not just the default branch.

---

**Owner \*** **Repository name \***


 muhammad200204 ▾ / study\_2023-2024\_arh-pc


✔ study\_2023-2024\_arh-pc is available.

Great repository names are short and memorable. Need inspiration? How about [animated-waffle](#) ?


**Description** (optional)

---

☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

---

 You are creating a public repository in your personal account.

---

[Create repository](#)

Рис. 4.12: Страница шаблона для репозитория

В открывшемся окне задаю имя репозитория (Repository name): study\_2022–

2023\_arh-pc и создаю репозиторий, нажимаю на кнопку «Create repository from template» (рис. 4.13).

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

*Required fields are marked with an asterisk (\*).*

### Repository template


 yamadharma/course-directory-student-template ▾

Start your repository with a template repository's contents.

☐ **Include all branches**

Copy all branches from yamadharma/course-directory-student-template and not just the default branch.

Owner \*

 muhammad200204 ▾

Repository name \*

/ study\_2023-2024\_arh-pc

✔ study\_2023-2024\_arh-pc is available.

Great repository names are short and memorable. Need inspiration? How about [animated-waffle](#) ?

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

 You are creating a public repository in your personal account.

Create repository

Рис. 4.13: Окно создания репозитория

Репозиторий создан (рис. 4.14).

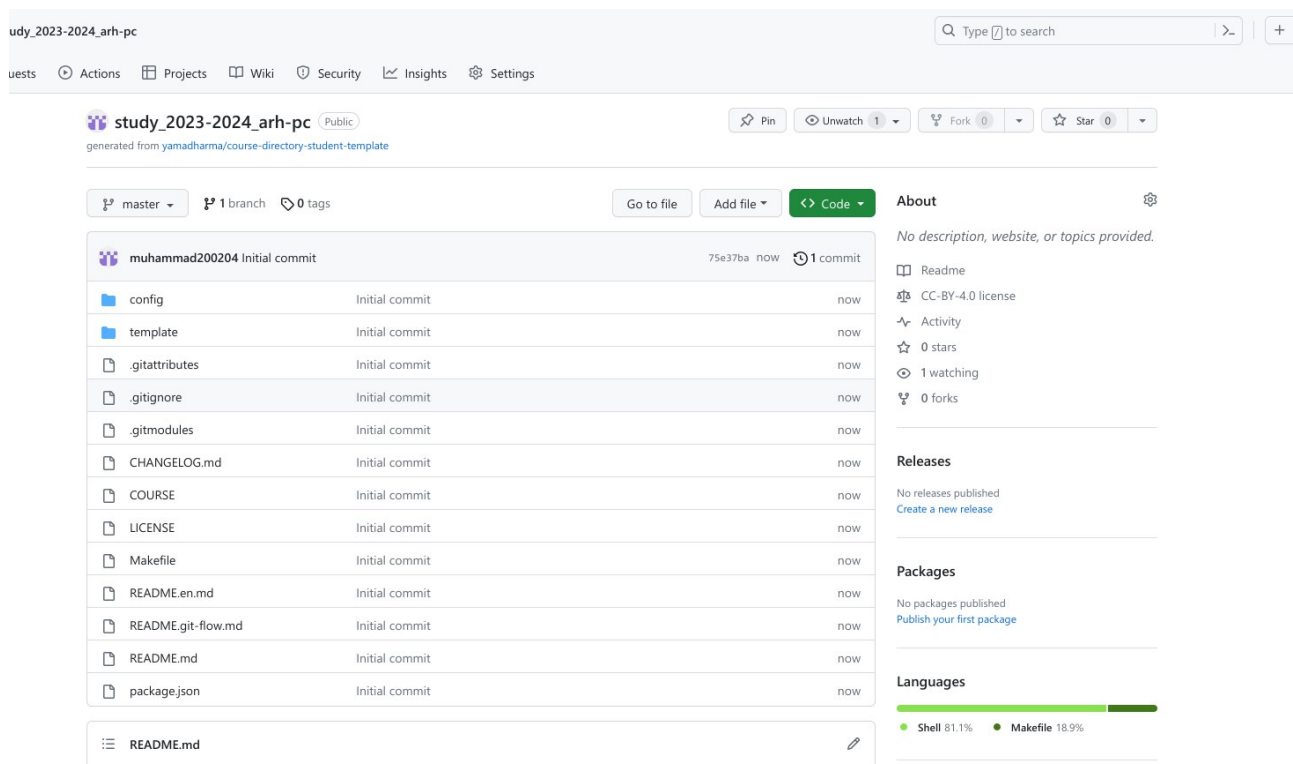


Рис. 4.14: Созданный репозиторий

Через терминал перехожу в созданный каталог курса с помощью утилиты `cd` (рис. 4.15)

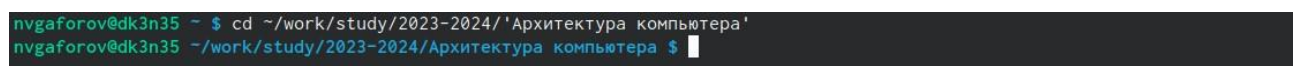


Рис. 4.15: Перемещение между директориями



Клонирую созданный репозиторий с помощью команды `git clone --recursive`

`git@github.com:/study_2022–2023_arh-pc.git arch-pc` (рис. 4.16).

```
nvgafarov@dk8n60 ~/work/study/2023–2024/Архитектура компьютера $ git clone --recursive git@github.com:muhammad200204/study_2023–2024_arh-pc.git arch-pc
Клонирование в «arch-pc»...
The authenticity of host 'github.com (140.82.121.3)' can't be established.
ED25519 key fingerprint is SHA256:+D1Y3wvvV6TuJhbpZisF/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: 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), 16.94 КиБ | 180.00 КиБ/с, 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 «/afs/.dk.sci.pfu.edu.ru/home/n/v/nvgafarov/work/study/2023–2024/Архитектура компьютера/arch-pc/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 КиБ | 1.18 МиБ/с, готово.
Определение изменений: 100% (28/28), готово.
Клонирование в «/afs/.dk.sci.pfu.edu.ru/home/n/v/nvgafarov/work/study/2023–2024/Архитектура компьютера/arch-pc/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 КиБ | 2.44 МиБ/с, готово.
Определение изменений: 100% (40/40), готово.
Submodule path 'template/presentation': checked out 'b1be3800ee91f5809264cb755d316174540b753e'
Submodule path 'template/report': checked out '1d1b61dcac9c287a83917b82e3aef11a33b1e3b2'
nvgafarov@dk8n60 ~/work/study/2023–2024/Архитектура компьютера $
```

Рис. 4.16: Клонирование репозитория

Копирую ссылку для клонирования на странице созданного репозитория, сначала перейдя в окно «code», далее выбрав в окне вкладку «SSH» (рис. 4.17).

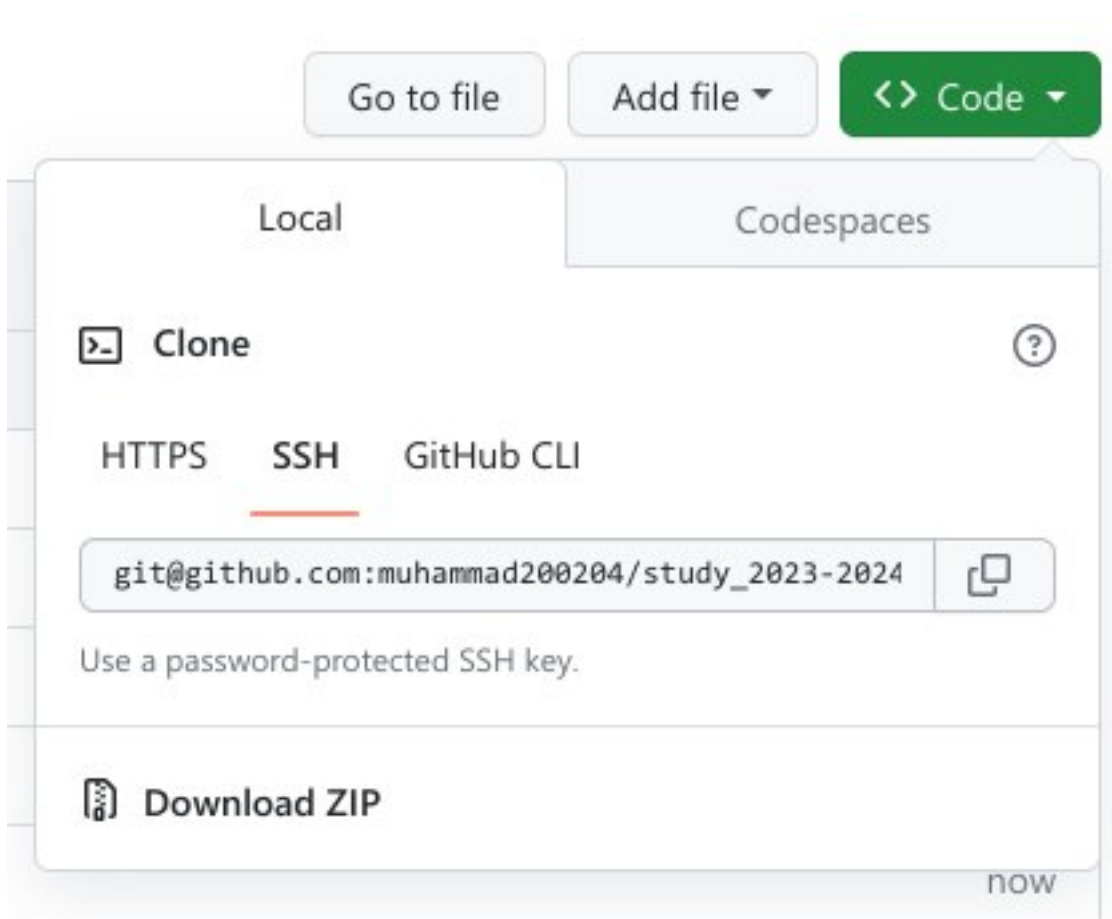


Рис. 4.17: Окно с ссылкой для копирования репозитория

#### 4.5 Настройка каталога курса

Перехожу в каталог `arch-pc` с помощью утилиты `cd` (рис. 4.18).

```
nvgaforov@dk8n60 ~/work/study/2023-2024/Архитектура компьютера $ cd ~/work/study/2023-2024/Архитектура компьютера/arch-pc
nvgaforov@dk8n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $
```

Рис. 4.18: Перемещение между директориями

Удаляю лишние файлы с помощью утилиты `rm` (рис. 4.19).

```
nvgaforov@dk8n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ rm package.json
nvgaforov@dk8n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $
```

Рис. 4.19: Удаление файлов

Создаю необходимые каталоги (рис. 4.20).

```
nvgaforov@dk8n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ echo arch-pc > COURSE
nvgaforov@dk8n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $
```

Рис. 4.20: Создание каталогов

Отправляю созданные каталоги с локального репозитория на сервер: добавляю все созданные каталоги с помощью `git add`, комментирую и сохраняю изменения на сервере как добавление курса с помощью `git commit` (рис. 4.21).

```
nvgaforov@dk8n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ git commit -am 'feat(main): make course structure'
[master 49b3c3b] feat(main): make course structure
199 files changed, 54725 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/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
create mode 100644 labs/lab03/presentation/Makefile
create mode 100644 labs/lab03/presentation/image/kulyabov.jpg
create mode 100644 labs/lab03/presentation/presentation.md
create mode 100644 labs/lab03/report/Makefile
create mode 100644 labs/lab03/report/bib/cite.bib
create mode 100644 labs/lab03/report/image/placeimg_800_600_tech.jpg
```

Рис. 4.21: Добавление и сохранение изменений на сервере

Отправляю все на сервер с помощью push (рис. 4.22).

```
nvgaforov@dk8n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ git push
Перечисление объектов: 37, готово.
Подсчет объектов: 100% (37/37), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (29/29), готово.
Запись объектов: 100% (35/35), 342.13 КиБ | 2.67 МиБ/с, готово.
Всего 35 (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:muhammad200204/study_2023-2024_arh-pc.git
   75e37ba..49b3c3b master -> master
nvgaforov@dk8n60 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $
```

Рис. 4.22: Выгрузка изменений на сервер

## 5 Выводы

При выполнении данной лабораторной работы я изучила идеологию и применение средств контроля версий, а также приобрела практические навыки по работе с системой git

## 6 Список литературы

1. Архитектура ЭВМ
2. Git - gitattributes Документация

