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: Куокконен Д.А.

Группа: НКАбд-03-23

Москва

2023 г.

Содержание

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. Сознание рабочего пространства и репозитория курса на основе шаблона
5. Сознание 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 (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. 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. Кроме того, 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

4.1 Настройка GitHub

Создаю учетную запись на сайте GitHub (рис. 4.1) Затем, я заполнила основные данные учетной записи. Тем самым, создала свой аккаунт на GitHub.

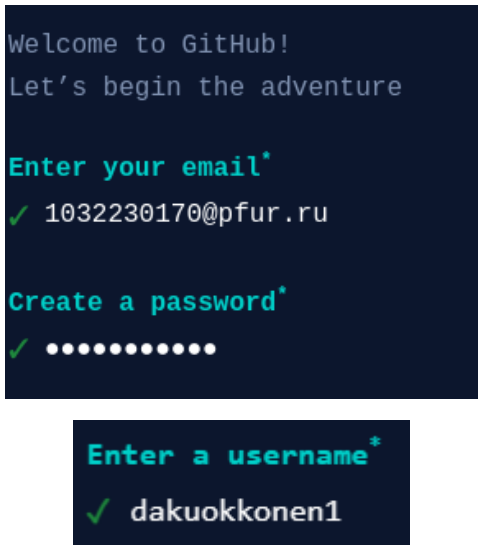

Рис.4.1 Заполнение данных учетной записи GitHub

Создание аккаунта

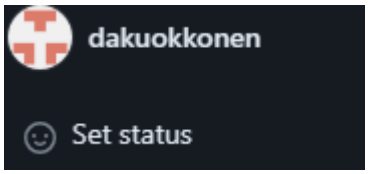

Рис.4.2 Аккаунт создан

4.2 Базовая настройка Git

Открывая 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 свою собственную (рис.4.3)

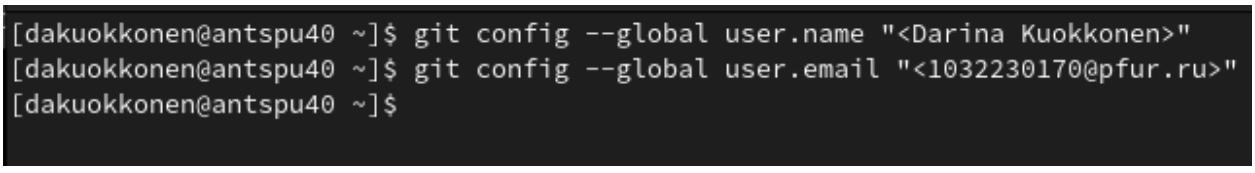

Рис.4.3 Предварительная конфигурация

Настраиваю urf-8 в выводе сообщений git для корректного отображения символов (рис.4.4)

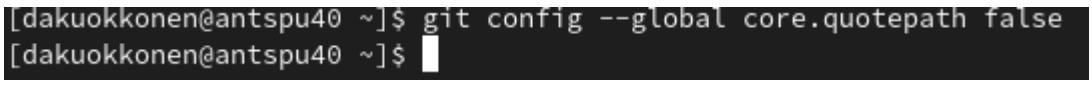

Рис.4.4 Настройка кодировки

Задаю имя “master” для начальной ветки (рис.4.5)

```
[dakuokkonen@antspu40 ~]$ git config --global init.defaultBranch master
[dakuokkonen@antspu40 ~]$
```

Рис.4.5 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. 4.6). CR и LF - это символы, которые можно использовать для обозначения разрыва строки в текстовых файлах.

```
[dakuokkonen@antspu40 ~]$ git config --global core.autocrlf input
[dakuokkonen@antspu40 ~]$
```

Рис. 4.6 Параметр autocrlf

Задаю параметр safecrIf со значением warn, так Git будет проверять преобразование на обратимость (рис. 4.7). При значении warn Git только выведет предупреждение, но будет принимать необратимые конвертации.

```
[dakuokkonen@antspu40 ~]$ git config --global core.safecrlf warn
[dakuokkonen@antspu40 ~]$
```

Рис. 4.7 Параметр safecrlf

4.3 Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозиториев 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. 4.8). Ключ автоматически сохранится в каталоге ~/.ssh/.

```
[dakuokkonen@antspu40 ~]$ ssh-keygen -C "Darina Kuokkonen <1032230170@pfur.ru>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dakuokkonen/.ssh/id_rsa):
Created directory '/home/dakuokkonen/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dakuokkonen/.ssh/id_rsa
Your public key has been saved in /home/dakuokkonen/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:R7EOdXEilgowbvZmUqjo+7JRc+j25Nh5vfEbO9m01Ww Darina Kuokkonen <1032230170@
pfur.ru>
The key's randomart image is:
+---[RSA 3072]----+
|    o.    =.+..  |
|   . o.  o.= o   |
|    = ....o      |
| . +.o  .+       |
|. .+..+ S o    ..|
|. o o+   .   . .E|
| o o .  .. .+ o. |
| .+ * .. .oooo   |
| o+o =.  ..+o    |
+----[SHA256]-----+
```

Рис 4.8 Генерация SSH-ключа

Xclip – утилита, позволяющая скопировать любой текст через терминал. Устанавливаю с помощью команды dnf install с ключом -y от имени суперпользователя, введя в начале команды sudo (рис.4.9)

```
[dakuokkonen@antspu40 ~]$ sudo dnf -y install xclip
[sudo] password for dakuokkonen:
Fedora 38 - x86_64                              3.2 MB/s |  83 MB     00:25
Fedora 38 openh264 (From Cisco) - x86_64        1.4 kB/s | 2.5 kB     00:01
Fedora Modular 38 - x86_64                      1.4 MB/s | 2.8 MB     00:01
Fedora 38 - x86_64 - Updates                    3.5 MB/s |  33 MB     00:09
Fedora Modular 38 - x86_64 - Updates            823 kB/s | 2.1 MB     00:02
Last metadata expiration check: 0:00:01 ago on Sat 30 Sep 2023 12:50:50 AM MSK.
Dependencies resolved.
================================================================================
 Package      Architecture  Version                        Repository     Size
================================================================================
Installing:
 xclip        x86_64        0.13-19.git11cba61.fc38        fedora         37 k

Transaction Summary
================================================================================
Install  1 Package

Total download size: 37 k
Installed size: 63 k
Downloading Packages:
xclip-0.13-19.git11cba61.fc38.x86_64.rpm        305 kB/s |  37 kB     00:00
--------------------------------------------------------------------------------
Total                                            53 kB/s |  37 kB     00:00
Fedora 38 - x86_64                              394 kB/s | 1.6 kB     00:00
Importing GPG key 0xEB10B464:
 Userid     : "Fedora (38) <fedora-38-primary@fedoraproject.org>"
 Fingerprint: 6A51 BBAB BA3D 5467 B617 1221 809A 8D7C EB10 B464
 From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-38-x86_64
Key imported successfully
Running transaction check
```

Рис.4.9 установка утилиты xclip

Копирую открытый ключ из директории, в которой он был сохранен, с помощью утилиты xclip (рис.4.10)

```
[dakuokkonen@antspu40 ~]$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
```

Рис.4.10 Копирование содержимого файла

Захожу на сайт GitHub. Открываю свой профиль и выбираю страницу “SSH and GPG keys”. Нажимаю кнопку “New SSH key”(рис.4.11)

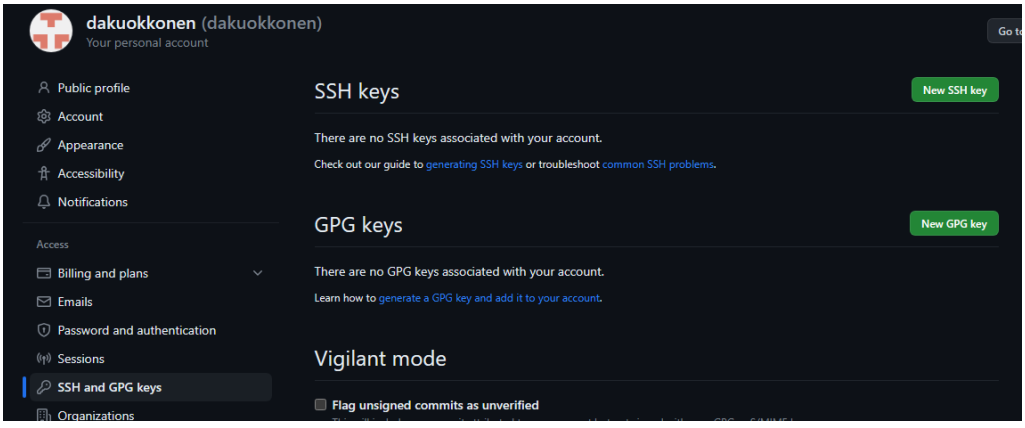

Рис.4.11 Окно SSH and GPG keys

Вставляю скопированный ключ в поле “Key”. В поле Title указываю имя для ключа. Нажимаю “Add SSH-key”, чтобы завершить добавление ключа (рис.4.12)

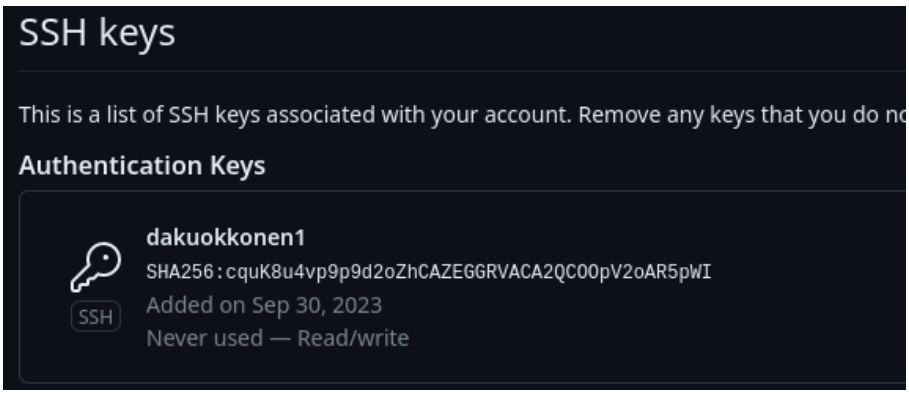

Рис.4.12 Добавление ключа

4.4 Создание рабочего пространства и репозитория курса на основе шаблона

Теперь я открываю терминал. Создаю директорию, рабочее пространство, с помощью утилиты mkdir, благодаря ключу -p создаю все директории после домашней -/work/study/2023-2024/"Архитектура компьютера" рекурсивно. И проверяю были ли созданы нужные мне каталоги. (рис.4.13)

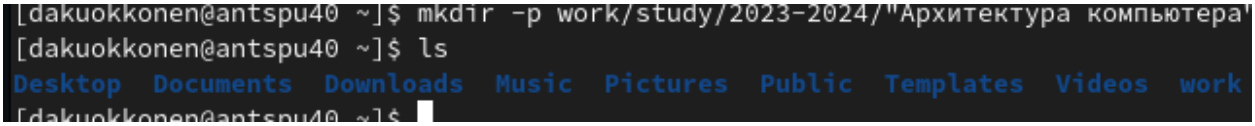

Рис. 4.13

4.5 Создание репозитория курса на основе шаблона

Перехожу на страницу репозитория. И выбираю “Use this template”, чтобы использовать этот шаблон для своего репозитория. (рис.4.14)

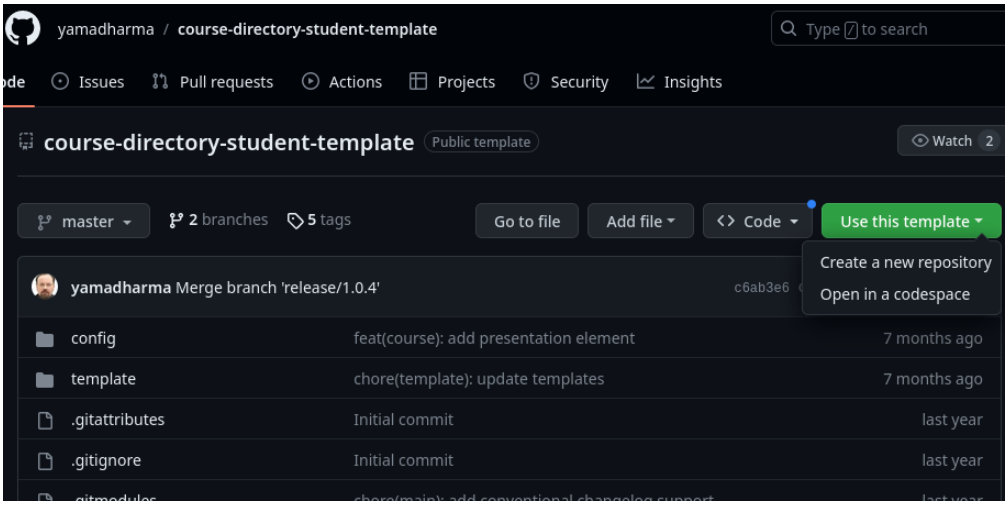

Рис. 4.14 Страница шаблона

Далее, задаю имя репозитория и создаю репозиторий. (рис.4.15)

Рис.4.15 Окно создания репозитория

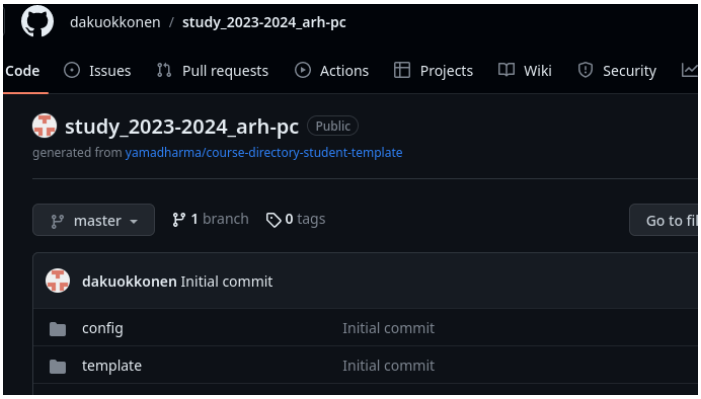

Рис.4.16 Репозиторий создан

Через терминал перехожу в созданный каталог курса (рис.4.17)

```
[dakuokkonen@antspu40 ~]$ cd ~/work/study/2023-2024/"Архитектура компьютера"
[dakuokkonen@antspu40 Архитектура компьютера]$
```

Рис.4.17 Перемещение между директориями

Затем, клонирую созданный мной репозиторий (рис.4.18)

```
[dakuokkonen@antspu40 Архитектура Компьютера]$ git clone --recursive https://git
hub.com/dakuokkonen/study_2023-2024_arh-pc.git arch-pc
Cloning into 'arch-pc'...
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
Receiving objects: 100% (27/27), 16.93 KiB | 270.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.
Submodule 'template/presentation' (https://github.com/yamadharma/academic-presen
tation-markdown-template.git) registered for path 'template/presentation'
Submodule 'template/report' (https://github.com/yamadharma/academic-laboratory-r
eport-template.git) registered for path 'template/report'
Cloning into '/home/dakuokkonen/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
Receiving objects: 100% (82/82), 92.90 KiB | 531.00 KiB/s, done.
Resolving deltas: 100% (28/28), done.
Cloning into '/home/dakuokkonen/work/study/2023-2024/Архитектура Компьютера/arch
-pc/template/report'...
remote: Enumerating objects: 101, done.
```

Рис.4.18 Клонирование

Копирую ссылку для клонирования на странице созданного репозитория (рис.4.19)

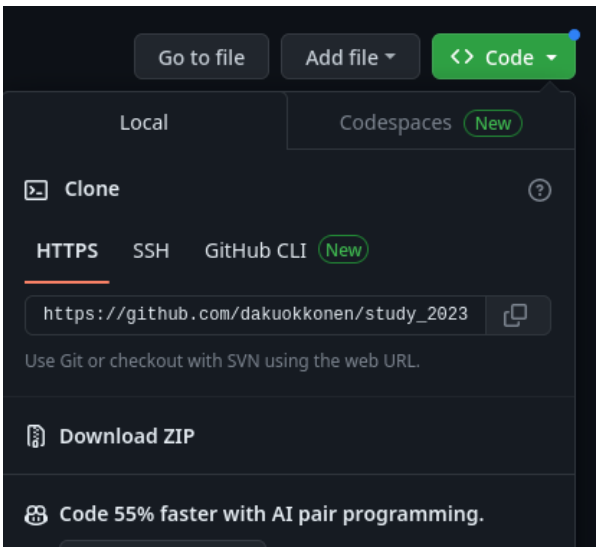

Рис.4.19 Окно для копирования

4.6 Настройка каталога курса

Для начала, перехожу в каталог с помощью утилиты

```
[dakuokkonen@antspu40 Архитектура Компьютера]$ cd ~/work/study/2023-2024/Архитек
тура\ Компьютера/arch-pc
```

Рис.4.20

Удаляю лишние файлы (рис.4.21)

```
[dakuokkonen@antspu40 arch-pc]$ rm package.json
```

Рис.4.21 Удаление лишних файлов

Теперь создадим необходимые каталоги (рис.4.22)

```
[dakuokkonen@antspu40 arch-pc]$ echo arch-pc > COURSE
[dakuokkonen@antspu40 arch-pc]$ make
[dakuokkonen@antspu40 arch-pc]$
```

Рис.4.22 Создание каталогов

Отправляю данные каталоги с локального репозитория на сервер, добавляю все созданные каталоги (рис.4.23)

```
[dakuokkonen@antspu40 arch-pc]$ git add .
[dakuokkonen@antspu40 arch-pc]$ git commit -am 'feat(main):make course structure
'
[master 32397e0] 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
```

Рис.4.23 Добавление и сохранение изменений на сервере

Отправляем все на сервер (рис.4.24)

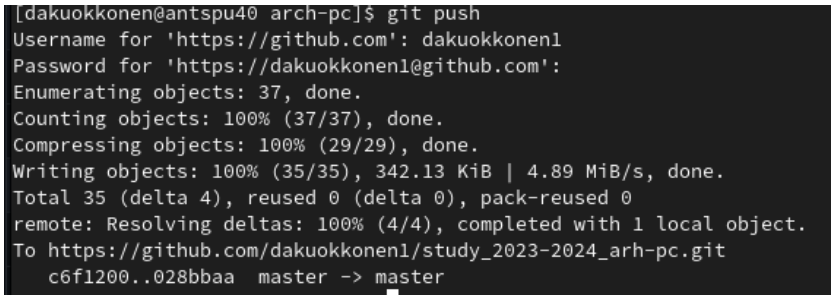

Рис.4.24 Выгрузка изменений на сервер

Проверяю правильность выполнения работы (рис.4.25)

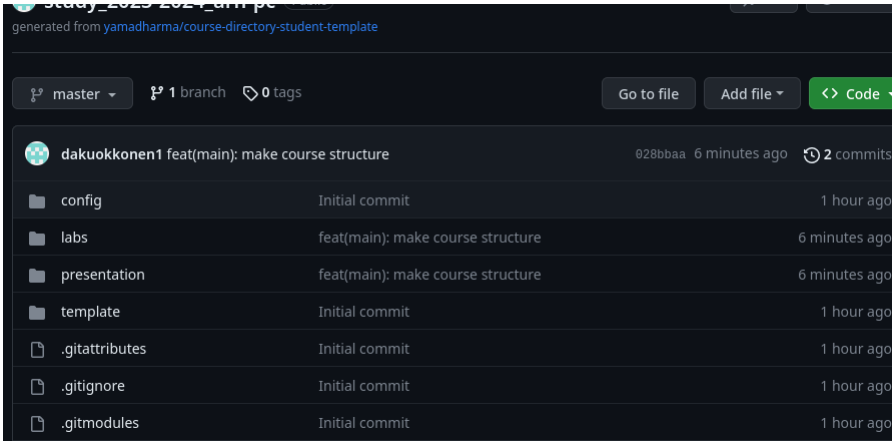

Рис.4.25 Проверка результата

4.7 Выполнение заданий для самостоятельной работы

1) Перехожу в директорию labs/lab02/report с помощью утилиты “cd”. И создаю в каталоге файл для отчета по третьей лабораторной работе. (рис.4.26)

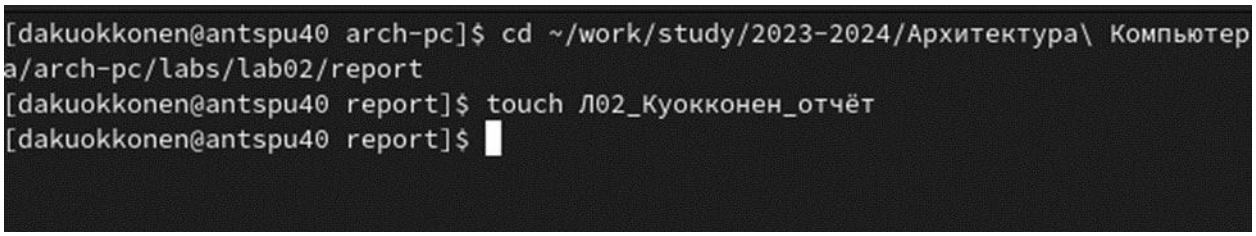

Рис.4.26 Создание файла

Оформить свой отчет я могу в текстовом процессоре LibreOffice writer, найдя его в меню приложений. После того, как я открыла его, открываю в нем созданный файл. (рис.4.27)

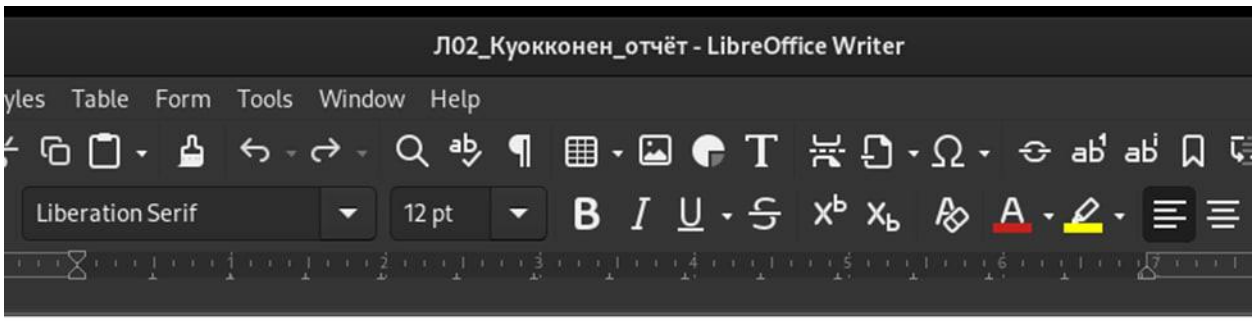

Рис.4.27 Работа с отчетом в текстовом редакторе

2) Проверяю местонахождение файла с отчетом по первой лабораторной работе. (рис.4.28)

```
[dakuokkonen@antspu40 ~]$ cd ~/Downloads
[dakuokkonen@antspu40 Downloads]$ ls
 Л01_Куокконен_отчёт.pdf  'Л02_Куокконен_отчет .pdf'
```

Рис. 4.28 Проверка местонахождения файла

Использую утилиту “cd”, чтобы попасть в подкаталог lab02/report, в котором находится отчет по первой лабораторной работе (рис.4.29)

```
[dakuokkonen@antspu40 report]$ cd ..
[dakuokkonen@antspu40 lab01]$ cd ..
[dakuokkonen@antspu40 labs]$ cd lab02
[dakuokkonen@antspu40 lab02]$ cd report
```

Рис.4.29

Добавляю файл Л01_Куокконен_отчёт (рис.4.30)

```
[dakuokkonen@antspu40 report]$ git add Л01_Куокконен_отчёт.pdf
```

Рис.4.30 Добавление файла на сервер

Теперь, сохраняем изменения на сервере командой git commit -m , поясняя, что добавила файлы.

То же самое делаю для отчета по второй лабораторной работе, перехожу в директорию labs/lab02/report с помощью утилиты cd, добавляю с помощью git add нужный мне файл, сохраняю изменения с помощью git commit (рис.4.31)

```
[dakuokkonen@antspu40 report]$ cd ~/work/study/2023-2024/Архитектура\ Компьютера
/arch-pc
[dakuokkonen@antspu40 arch-pc]$ cd labs/lab02/report
[dakuokkonen@antspu40 report]$ git add Л02_Куокконен_отчет.pdf
```

Рис. 4.31 Подкаталоги и файлы в репозитории

Отправляю данный файл в центральный репозиторий, с помощью команды git push -f origin master и проверяю на сайте GitHub правильность выполнения заданий. (рис.4.32, 4.33,4.34)

```
[dakuokkonen@antspu40 report]$ git push -f origin master
```

Рис.4.32

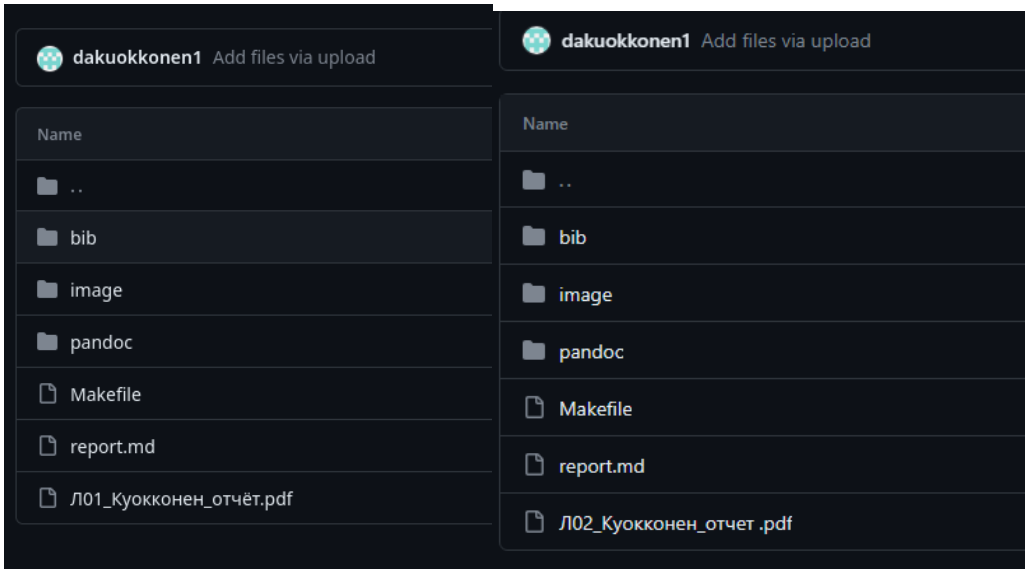

Рис 4.33,4.34 Страницы каталога в репозитории

5 Вывод

Я приобрела практический опыт работы с системой Git, изучила принципы и применение контроля версий.

6 Список литературы

1. Git (github.com)

2. Пользовательское соглашение (rudn.ru)