РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1.

дисциплина: Архитектура компьютера

Студент: Гаязов Рузаль

Группа: НКАбд-04-24

МОСКВА

2024 г.

1. Цель

Целью работы является изучить идеологию и применение средств контроля версий. Приобрести практические навыки по работе с системой git.

2. Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при ра- боте нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен до- ступ 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 заблокиро вать 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 измене-

ний 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.

3. Лабораторная работа

Настройка GitHub

Создаю учетную запись на сайте GitHub. Далее я заполнил основные данные учетной записи. Аккаунт создан рис(4.1)

Рис 4.1: Аккаунт создан

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).

```
liveuser@localhost-live:~$ git config --global user.name "<Ruzal Gayazov>"
liveuser@localhost-live:~$ git config --global user.email "<lol300005@outlook.com
```

Рис 4.2: Предварительная конфигурация git

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

```
> git config --global core.quotepath false
>
```

Рис. 4.3: Настройка кодировки

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

```
> git config --global init.defualtBranch 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.

```
> git config --global core.autoctlf 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.

```
> git config --global core.safecrlf warn
```

Рис. 4.6: Параметр safecrlf

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. 4.7). Ключ автоматически сохранится в каталоге ~/.ssh/.

```
liveuser@localhost-live:~$ ssh-keygen -C "Ruzal Gayazov <lol300005@outlook.com>"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/liveuser/.ssh/id_ed25519):
Created directory '/home/liveuser/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/liveuser/.ssh/id_ed25519
Your public key has been saved in /home/liveuser/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:sAmDblSvD5zOVbT7iLpcL/hA4qU/87BPuA6up07muCE Ruzal Gayazov <lol300005@outlook.com>
The key's randomart image is:
+--[ED25519 256]--+
|    .   .        |
|   o . . .       |
|  o o o o        |
| o . = = .       |
|  + B + S        |
| o B = . o       |
|Eo+ *.= . .      |
|*o.++O..         |
|*B..OB+..        |
+----[SHA256]-----+
liveuser@localhost-live:~$
liveuser@localhost-live:~$
```

Рис. 4.7: Генерация SSH-ключа

Xclip – утилита, позволяющая скопировать любой текст через терминал. Оказывается, в дистрибутиве Linux Kali ее сначала надо установить. Устанавливаю xclip с помощью команды dnf install с ключом -y отимени суперпользователя, введя в начале команды sudo (рис. 4.8).

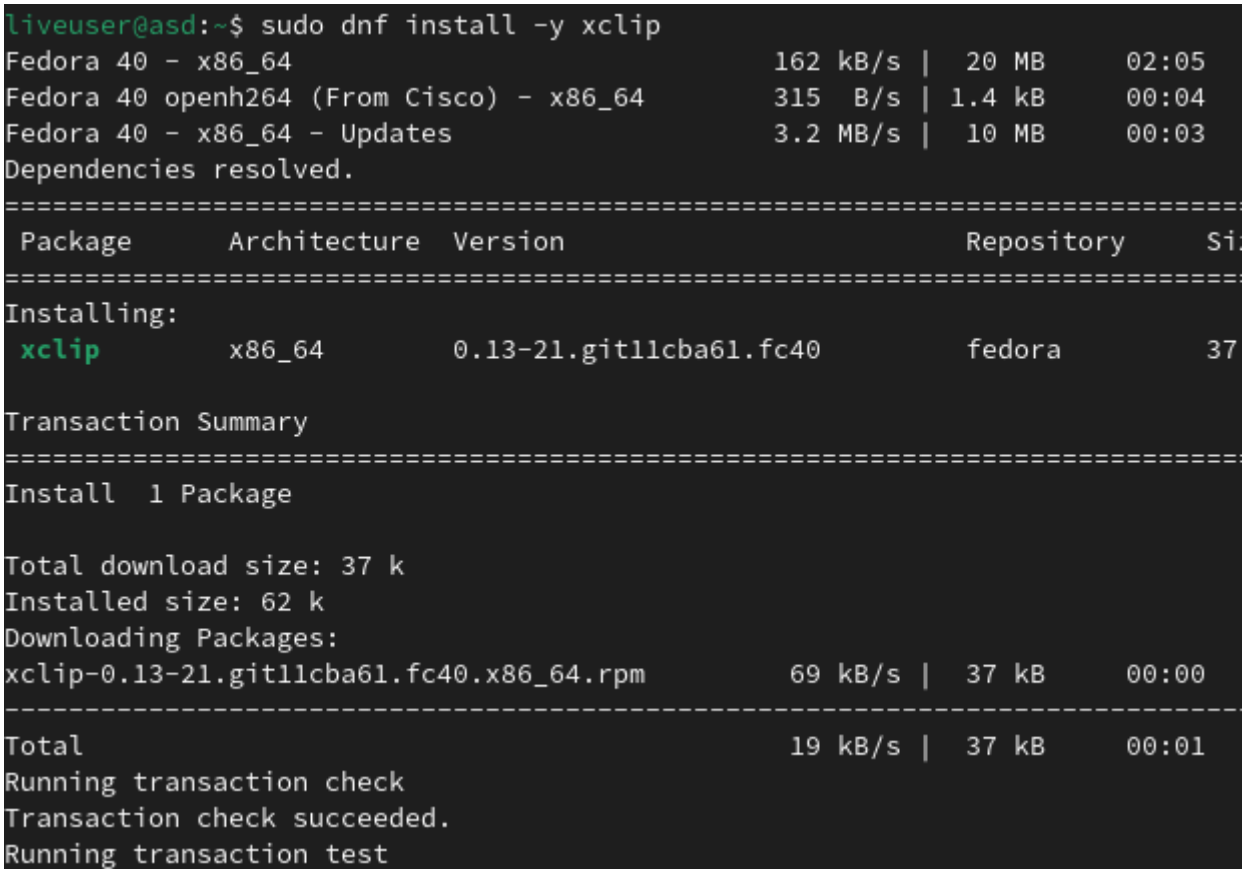

Рис. 4.8: Установка утилиты xclip

Копирую открытый ключ из директории, в которой он был сохранен, с помощью утилиты xclip (рис. 4.9).

Рис. 4.9: Копирование содержимого файла

Открываю браузер, захожу на сайт GitHub. Открываю свой профиль и выбираю страницу «SSH and GPG keys». Нажимаю кнопку «New SSH key» . Вставляю скопированный ключ в поле «Key». В поле Title указываю имя для ключа. Нажимаю «Add SSH-key», чтобы завершить добавление ключа (4.10).

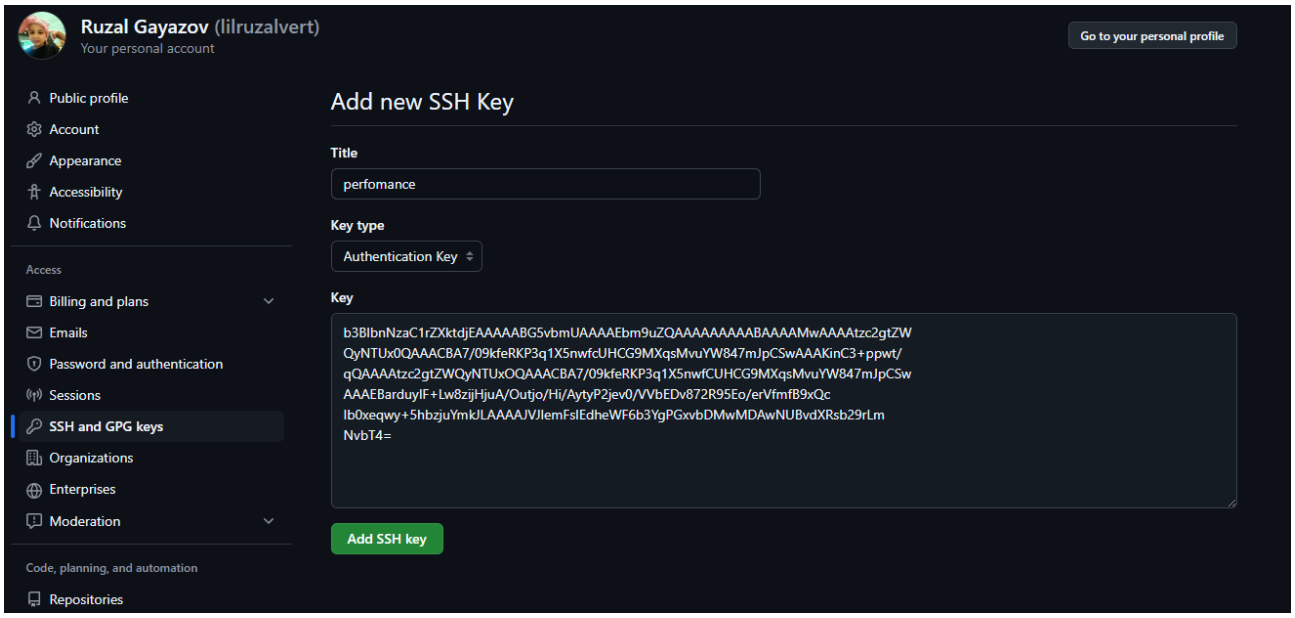

Рис. 4.10: Добавление ключа

Создание рабочего пространства и репозитория курса на основе шаблона.

Создаю директорию, рабочее пространство, с помощью утилиты mkdir, блягодаря 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).

```
liveuser@asd:~$ cat /home/liveuser/.ssh/id_ed25519.pub | xclip -sel clip
liveuser@asd:~$ mkdir -p work/study/2022-2023/"Архитектура компьютера"
liveuser@asd:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos  work
liveuser@asd:~$
```

Рис. 4.11: Создание рабочего пространства

Создание репозитория курса на основе шаблона

В браузере перехожу на страницу репозитория с шаблоном курса по адресу https://github.com/yamadharma/course-directory-student-template. Далее выбираю «Use this template», чтобы использовать этот шаблон для своего репозитория (рис. 4.12).

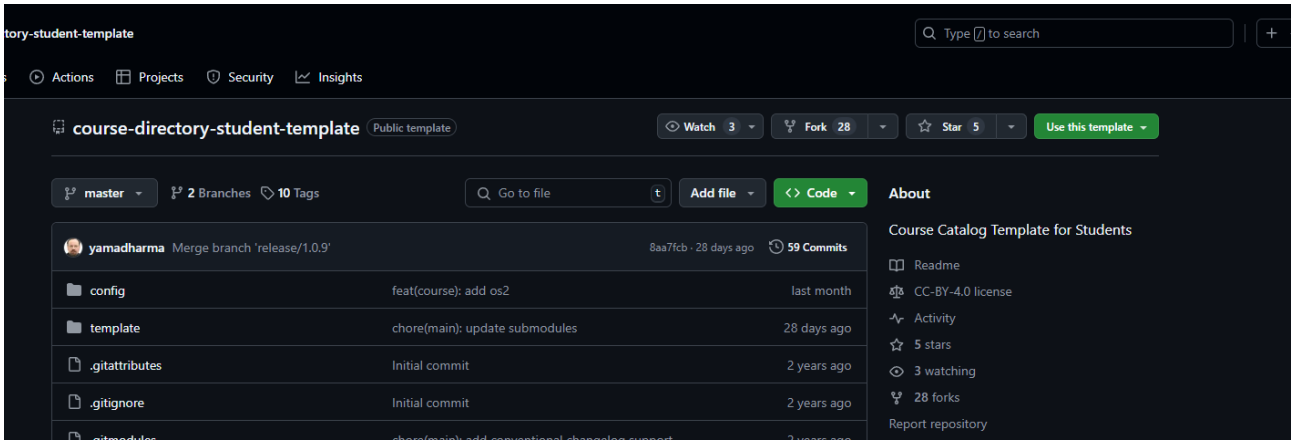

Рис. 4.12: Страница шаблона для репозитория

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).

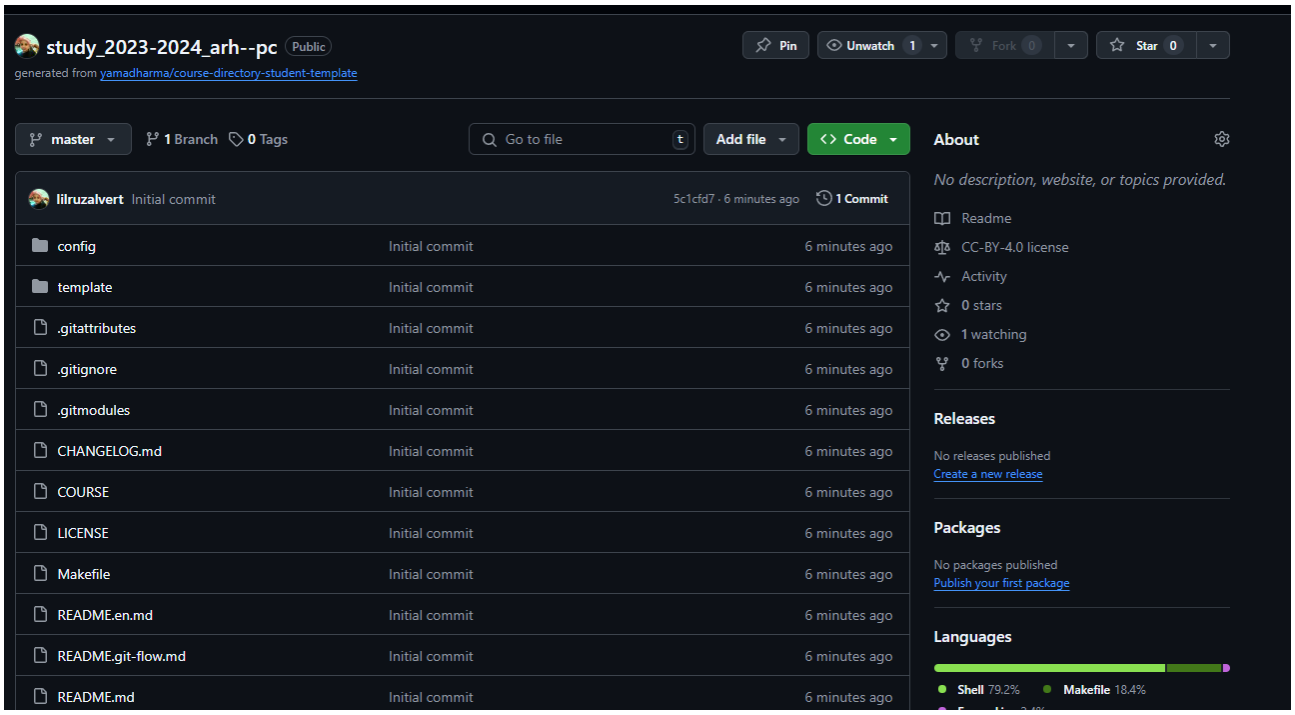

Рис. 4.13: Репозиторий

Репозиторий создан. Через терминал перехожу в созданный каталог курса с помощью утилиты cd. Клонирую созданный репозиторий с помощью команды git clone –recursive git@github.com:/study_2023–

2024_arh-pc.git arch-pc (рис. 4.14).

```
liveuser@asd:~/work/study/2023-2024/Архитектура компьютера$ git clone --recursive git@github.com:lilruzalvert/study_2023-2024_arh-pc.git arch-pc
Cloning into 'arch-pc'...
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (33/33), done.
```

Рис. 4.14: Клонирование репозитория

Копирую ссылку для клонирования на странице созданного репозитория, сначала перейдя в окно «code», далее выбрав в окне вкладку «SSH» (рис. 4.15).

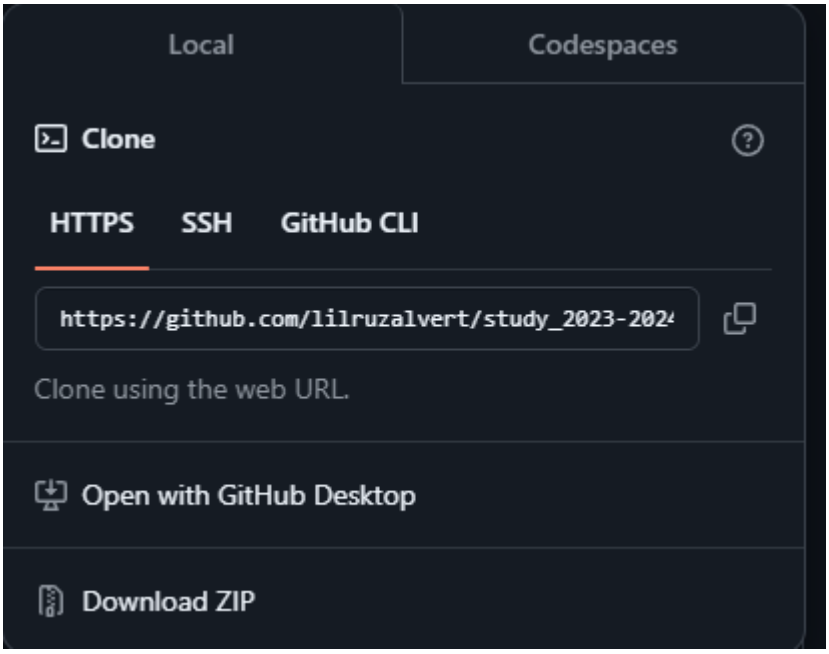

Рис. 4.15: Окно с ссылкой для копирования репозитория

Настройка каталога курса

Перехожу в каталог arch-pc с помощью утилиты cd. Удаляю лишние файлы с помощью утилиты rm (Рис 4.16).

```
liveuser@asd:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ rm package.json
```

Рис. 4.16: Удаление файлов

Создаю необходимые каталоги (рис. 4.17).

```
liveuser@asd:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ echo arch-pc > COURSE
```

Рис. 4.17: 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. 4.18).

```
liveuser@asd:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git commit -am 'feat(main): make course structure'
[master 3c0a0c2] feat(main): make course structure
 2 files changed, 1 insertion(+), 14 deletions(-)
 delete mode 100644 package.json
```

Рис. 4.18: Добавление и сохранение изменений на сервере

Отправляю все на сервер с помощью push (рис. 4.19).

```
liveuser@asd:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ git push



Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 5 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 300 bytes | 300.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:lilruzalvert/study_2023-2024_arh-pc.git
   5c1cfd7..3c0a0c2  master -> master
```

Рис. 4.19: Выгрузка изменений на сервер

Проверяю правильность выполнения работы сначала на самом сайте GitHub (рис. 4.20).

Рис. 4.21: Страница репозитория

4. Выводы

В результате выполнения данной лабораторной работы я изучил идеологию и применение средств контроля версий, а также приобрел практические навыки по работе с системой git.

5. Список литературы

Лабораторная работа №2 (Архитектура ОС).
Архитектура ЭВМ
Git - gitattributes Документация