

Лабораторная работа №2

Дисциплина Архитектура компьютера

Алехин Давид Андреевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
5	Выводы	16
	Список литературы	17

Список иллюстраций

4.1	Пункт 2.4.2	10
4.2	Пункт 2.4.3	10
4.3	Пункт 2.4.3	11
4.4	Пункт 2.4.3	11
4.5	Пункт 2.4.3	12
4.6	Пункт 2.4.4	12
4.7	Пункт 2.4.5	12
4.8	Пункт 2.4.5	13
4.9	Пункт 2.4.5 продолжение	13
4.10	Пункт 2.4.5 продолжение	13
4.11	Пункт 2.4.6	14
4.12	Пункт 2.4.6	14
4.13	Пункт 2.4.6	15
4.14	Пункт 2.5	15

Список таблиц

1 Цель работы

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

2 Задание

1.Базовая настройка git. 2.Создание рабочего пространства и репозитория курса на основе шаблона. 3.Настройка каталога курса. 4.Добавление файлов с 1 и 2 лабораторной в папки report.

3 Теоретическое введение

2.2.1. Системы контроля версий. Общие понятия 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 заблоки-

ровать 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.

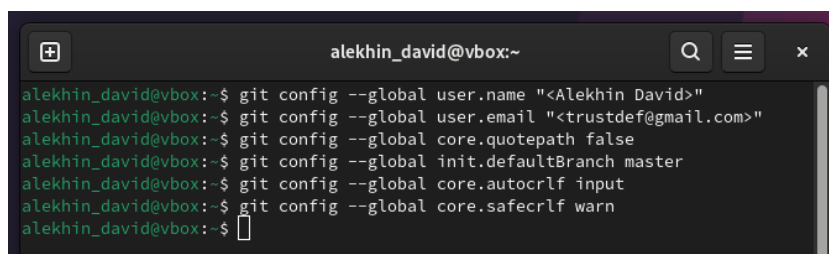
2.2.2. Система контроля версий Git репозитория Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` с различными опциями. Благодаря тому, что Git является распределённой системой контроля версий, резервную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

2.2.4. Стандартные процедуры работы при наличии центрального репозитория 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): `git checkout master` `git pull` `git checkout -b имя_ветки` 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.

Для этого необходимо проверить, какие файлы изменились к текущему моменту: `git status` и при необходимости удаляем лишние файлы, которые не хотим отправлять в центральный репозиторий. Затем полезно просмотреть текст изменений на предмет соответствия правилам ведения чистых коммитов: `git diff`. Если какие-либо файлы не должны попасть в коммит, то помечаем только те файлы, изменения которых нужно сохранить. Для этого используем команды добавления и/или удаления с нужными опциями: `git add имена_файлов` `git rm имена_файлов`. Если нужно сохранить все изменения в текущем каталоге, то используем: `git add .` Затем сохраняем изменения, поясняя, что было сделано: `git commit -am "Some commit message"` и отправляем в центральный репозиторий: `git push origin имя_ветки` или `git push`.

4 Выполнение лабораторной работы

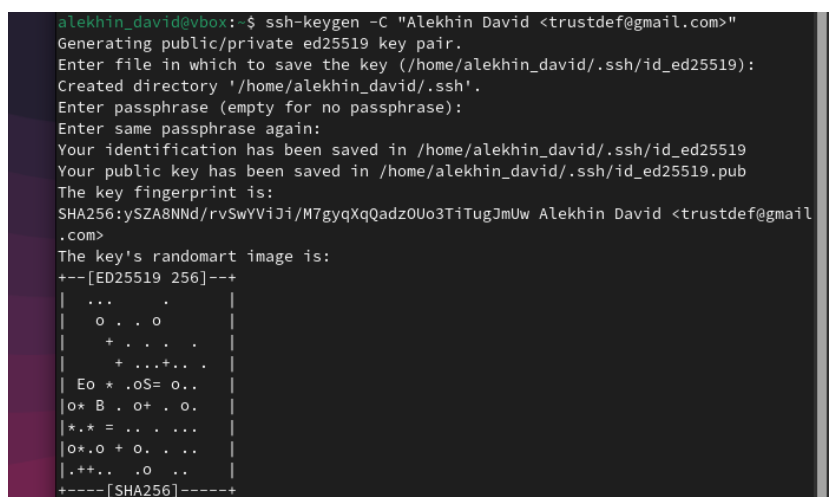
1.Выполняю пункт 2.4.2. (Базовая настройка git): Создаю учётную запись и создаю предварительную конфигурацию git, после настраиваю utf-8 в выводе сообщений git, задаю имя начальной ветки master и ввожу параметр autocrlf и safecrlf (рис. 4.1).

A terminal window titled 'alekhin_david@vbox:~' with search, menu, and close icons. It shows a series of git config commands being executed to set up a global user profile and repository settings.

```
alekhin_david@vbox:~$ git config --global user.name "Alekhin David"
alekhin_david@vbox:~$ git config --global user.email "<trustdef@gmail.com>"
alekhin_david@vbox:~$ git config --global core.quotepath false
alekhin_david@vbox:~$ git config --global init.defaultBranch master
alekhin_david@vbox:~$ git config --global core.autocrlf input
alekhin_david@vbox:~$ git config --global core.safecrlf warn
alekhin_david@vbox:~$
```

Рис. 4.1: Пункт 2.4.2

2.Выполняю пункт 2.4.3. (Создание SSH ключа) Генерирую пару ключей далее генерирую открытый ключ (рис. [4.2]4.3[4.4]4.5).

A terminal window showing the execution of the ssh-keygen command to generate an ED25519 key pair. It displays the directory creation, passphrase prompts, and the resulting key fingerprint and randomart image.

```
alekhin_david@vbox:~$ ssh-keygen -C "Alekhin David <trustdef@gmail.com>"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/alekhin_david/.ssh/id_ed25519):
Created directory '/home/alekhin_david/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/alekhin_david/.ssh/id_ed25519
Your public key has been saved in /home/alekhin_david/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:ySZA8NND/rvSwYVji/M7gyqXqQadz0Uo3TiTugJmUw Alekhin David <trustdef@gmail.com>
The key's randomart image is:
+--[ED25519 256]--+
|  . . .          |
|  o . . o        |
|    + . . .      |
|    + . . . .    |
|  Eo * .oS= o..  |
|o* B . o+ . o.   |
|*. * = . . . .   |
|o*.o + o. . . .  |
|,++.. .o . .     |
+----[SHA256]-----+
```

Рис. 4.2: Пункт 2.4.3

```
alekhin_david@vbox:~$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
cat: /home/alekhin_david/.ssh/id_rsa.pub: Нет такого файла или каталога
bash: xclip: команда не найдена...
Установить пакет «xclip», предоставляющий команду «xclip»? [N/y] y

* Ожидание в очереди...
* Загрузка списка пакетов...
Следующие пакеты должны быть установлены:
xclip-0.13-21.git11cba61.fc40.x86_64 Command line clipboard grabber
Продолжить с этими изменениями? [N/y] y

* Ожидание в очереди...
* Ожидание аутентификации...
* Ожидание в очереди...
* Загрузка пакетов...
* Запрос данных...
* Проверка изменений...
* Установка пакетов...

alekhin_david@vbox:~$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
cat: /home/alekhin_david/.ssh/id_rsa.pub: Нет такого файла или каталога
alekhin_david@vbox:~$ ls -la
.                .pki
..               .ssh
.bash_history    tmp
.bash_logout     .var
.bash_profile    .vboxclient-clipboard-tty2-control.pid
.bashrc          .vboxclient-clipboard-tty2-service.pid
.cache           .vboxclient-draganddrop-tty2-control.pid
.config          .vboxclient-draganddrop-tty2-service.pid
.gitconfig       .vboxclient-hostversion-tty2-control.pid
.local           .vboxclient-seamless-tty2-control.pid
.mozilla         .vboxclient-seamless-tty2-service.pid
alekhin_david@vbox:~$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
cat: /home/alekhin_david/.ssh/id_rsa.pub: Нет такого файла или каталога
alekhin_david@vbox:~$ cat -a ~/.ssh/id_rsa.pub | xclip -sel clip
cat: неверный ключ - «a»
По команде «cat --help» можно получить дополнительную информацию.
alekhin_david@vbox:~$ cd .ssh
bash: cd: .ssh: Нет такого файла или каталога
alekhin_david@vbox:~$ cd -a .ssh
bash: cd: -a: недопустимый параметр
cd: использование: cd [-L|[-P [-e]] [-@]] [каталог]
```

Рис. 4.3: Пункт 2.4.3

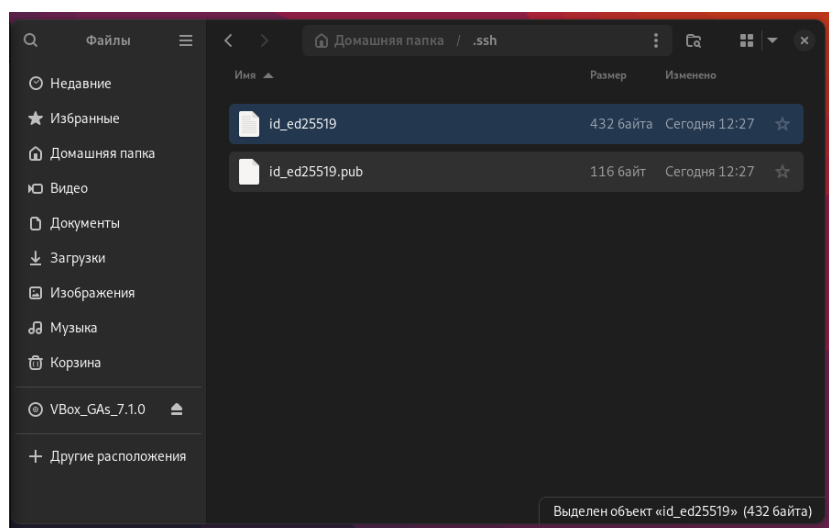


Рис. 4.4: Пункт 2.4.3

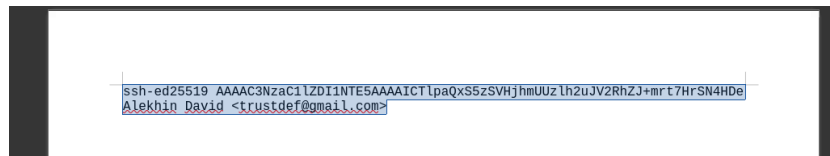


Рис. 4.5: Пункт 2.4.3

3. Изучаю структуру рабочего пространства и выполняю пункт 2.4.4. (Создание рабочего пространства и репозитория курса на основе шаблона) Создаю каталог для предмета Архитектура компьютера (рис. 4.6).

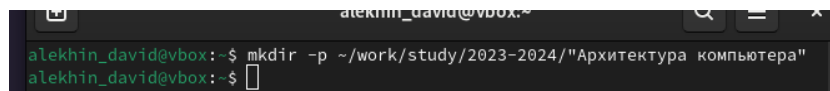


Рис. 4.6: Пункт 2.4.4

4. Выполняю пункт 2.4.5. (Создание репозитория курса на основе шаблона), создаю репозиторий фото 7-8. (рис. [4.7]4.8).

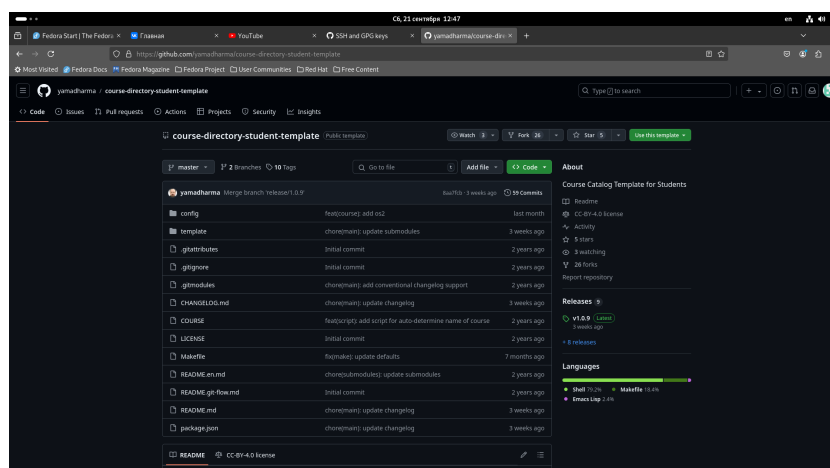


Рис. 4.7: Пункт 2.4.5

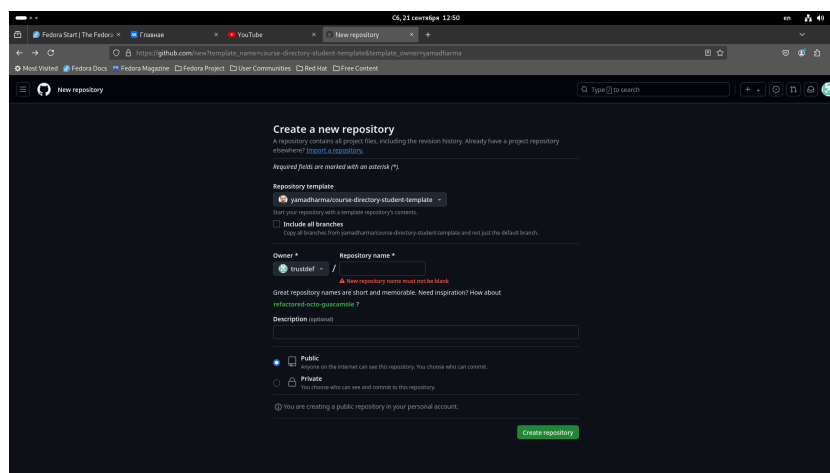


Рис. 4.8: Пункт 2.4.5

5. Далее клонирую созданный репозиторий в ранее созданный каталог (почему-то не получается перейти в каталог одной командой, перехожу с помощью последовательного перемещения по каталогу с помощью команды `cd`), (рис. [4.9]4.10).

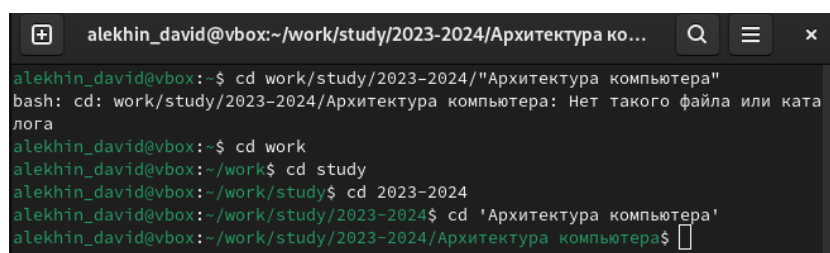


Рис. 4.9: Пункт 2.4.5 продолжение

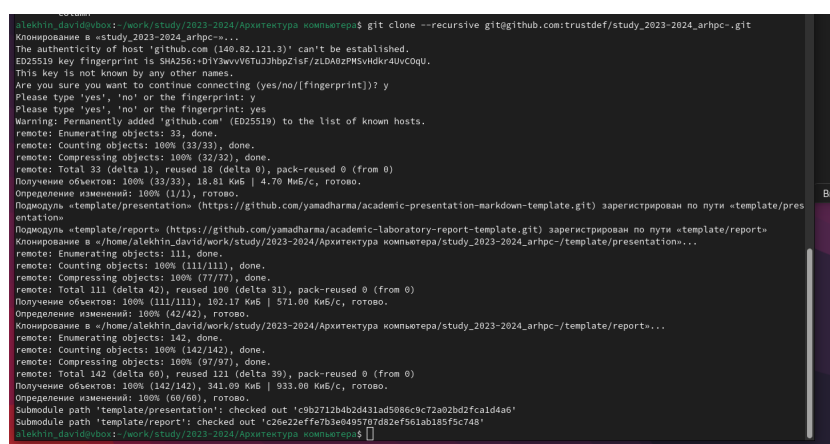


Рис. 4.10: Пункт 2.4.5 продолжение

6.Выполняю пункт 2.4.6. (Настройка каталога курса) Удаляю лишние файлы, добавляю нужные и отправляю их на сервер (рис. 4.11).Проверяю правильность создания (рис. [4.12]4.13).

```
alekhn.david@vbox: /work/study/2023-2024/Архитектура_компьютера$ cd study_2023-2024_arhpc-
alekhn.david@vbox: /work/study/2023-2024/Архитектура_компьютера/study_2023-2024_arhpc$ rm package.json
alekhn.david@vbox: /work/study/2023-2024/Архитектура_компьютера/study_2023-2024_arhpc$ echo arch-pc > COURSE
make
Usage:
make <target>

Targets:
list                List of courses
prepare            Generate directories structure
submodule          Update submodules

alekhn.david@vbox: /work/study/2023-2024/Архитектура_компьютера/study_2023-2024_arhpc$ git add .
alekhn.david@vbox: /work/study/2023-2024/Архитектура_компьютера/study_2023-2024_arhpc$ git commit -am 'feat(main): make course structure'
[master bb96201] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
alekhn.david@vbox: /work/study/2023-2024/Архитектура_компьютера/study_2023-2024_arhpc$ git push
Перечисление объектов: 5, готово.
Подсчет объектов: 100% (5/5), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (2/2), готово.
Запись объектов: 100% (3/3), 287 байтов | 287.00 КиБ/с, готово.
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:trustdef/study_2023-2024_arhpc-.git
 bd9b5e0..bb96201 master -> master
alekhn.david@vbox: /work/study/2023-2024/Архитектура_компьютера/study_2023-2024_arhpc$
```

Рис. 4.11: Пункт 2.4.6

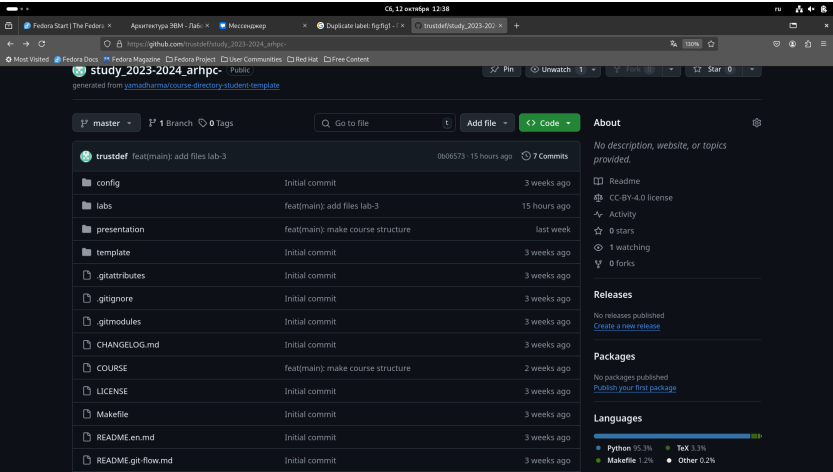


Рис. 4.12: Пункт 2.4.6

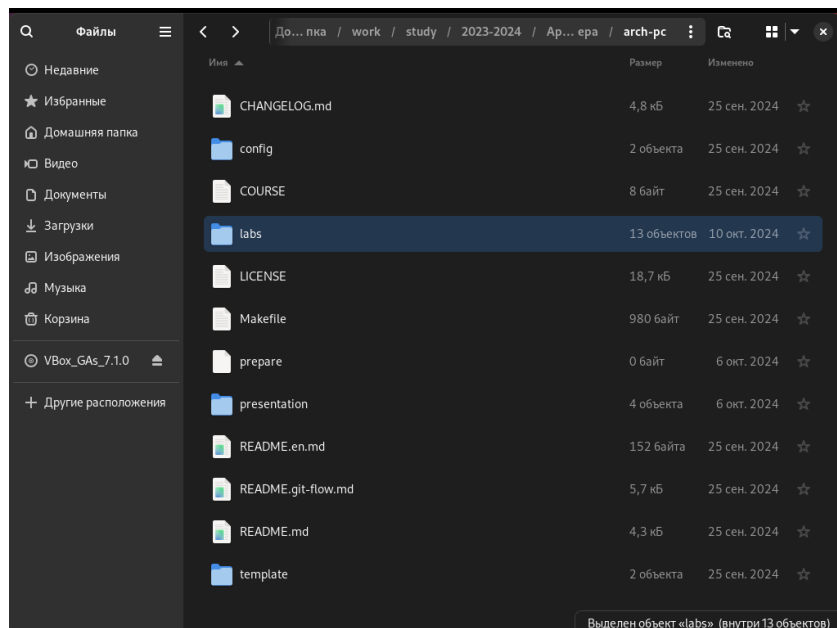


Рис. 4.13: Пункт 2.4.6

7.Выполняю пункт 2.5: Добавляю файлы с 1 и 2 лабораторной в папки labs/lab01, labs/lab02 в рабочее пространство git и отправляю их на сервер (рис. 4.14).

```

alekhin_david@vbox: ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc $ git add .
alekhin_david@vbox: ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc $ git commit -am 'feat(main): make course structure'
[master fcb9281] feat(main): make course structure
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 labs/lab1/report1.odt
create mode 100644 labs/lab2/report2.odt
alekhin_david@vbox: ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc $ git push
Перечисление объектов: 8, готово.
Подсчет объектов: 100% (8/8), готово.
При сжатии изменения используются до 6 потоков
Сжатие объектов: 100% (5/5), готово.
Запись объектов: 100% (7/7), 2,27 МБ | 997,00 КиБ/с, готово.
Total 7 (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:trustdef/study_2023-2024_arhpc-.git
   b99281..fcb9281 master -> master
alekhin_david@vbox: ~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arhpc $

```

Рис. 4.14: Пункт 2.5

5 Выводы

Проведя лабораторную работу я научился работать с системой git, создал в этой системе рабочее пространство и настроил его. А также изучил идеологию и применение средств контроля версий.

Список литературы

::: Файл с лабораторной работой 2 и прилагающиеся к нему материалы: 1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>. 2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>. 3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>. 4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>. 5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>. 6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591. :::