

Отчёт по лабораторной работе №2

дисциплина: Архитектура компьютера

Курушин Георгий Романович

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Настройка GitHub	9
4.2	Базовая настройка git	9
4.3	Создание SSH ключа	10
4.4	Создание рабочего пространства, репозитория курса на основе шаблона	12
4.5	Создание репозитория курса на основе шаблона	12
4.6	Настройка каталога курса	14
4.7	Выполнение заданий для самостоятельной работы	15
5	Выводы	18
	Список литературы	19

Список иллюстраций

4.1	Учетная запись на GitHub.	9
4.2	Предварительная конфигурация в git.	9
4.3	Настраивание utf-8.	10
4.4	Имя начальной ветки.	10
4.5	Ввод команд autocrlf и safecrlt.	10
4.6	Генерация ключей.	10
4.7	Скачивание команды xclip.	11
4.8	Копирование ключа.	11
4.9	Вставка ключа.	12
4.10	Создание терминала для предмета «Архитектура компьютера». . .	12
4.11	Выбор шаблона	13
4.12	Создание репозитория.	13
4.13	Переход в каталог курса	13
4.14	Клонирование репозитория.	14
4.15	Удаление лишних файлов в каталоге курса.	14
4.16	Создание каталогов и их отправка на сервер.	14
4.17	Проверка выполненной работы.	15
4.18	Создание отчета о выполнении работы.	15
4.19	Местонахождение лабораторных работ.	15
4.20	Копирование отчета по лабораторной работе в нужный каталог. . .	16
4.21	Добавление файлов с помощью команды git add.	16
4.22	Поручаю консоли совершить изменения.	16
4.23	Команды git status и git push для завершения копирования.	17
4.24	Проверка проделанных операций.	17

Список таблиц

1 Цель работы

Целью работы является применение средств контроля версий. А также очень важно 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

Система контроля версий (Version Control System, VCS) — это инструмент, используемый разработчиками программного обеспечения для управления изменениями в исходном коде и других файловых ресурсах. Системы контроля версий разработаны специально для того, чтобы максимально упростить и упорядочить работу над проектом (вне зависимости от того, сколько человек в этом участвуют). СКВ дает возможность видеть, кто, когда и какие изменения вносил; позволяет формировать новые ветви проекта, объединять уже имеющиеся; настраивать контроль доступа к проекту ; осуществлять откат до предыдущих версий. 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. Демидова А. В. 14 Архитектура ЭВМ 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 внёс. 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.

4 Выполнение лабораторной работы

4.1 Настройка GitHub

Для выполнения лабораторной работы создаю учетную запись на <https://github.com/>

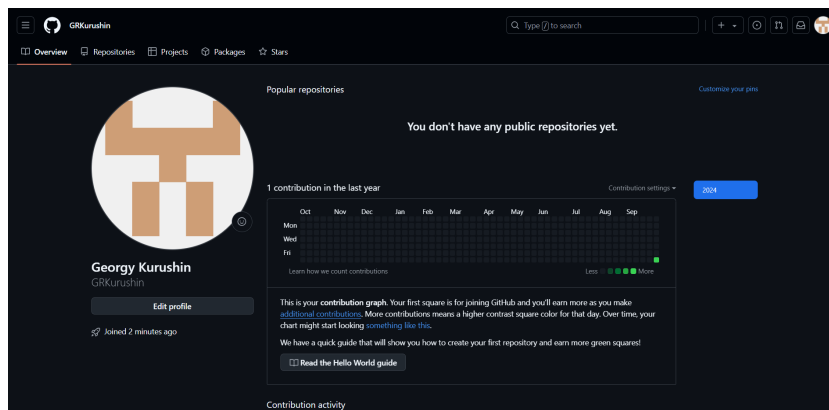


Рис. 4.1: Учетная запись на GitHub.

4.2 Базовая настройка git

Делаю предварительную конфигурацию git. Захожу в терминал и ввожу команды, указывая свое имя и email

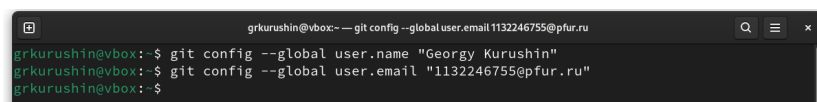


Рис. 4.2: Предварительная конфигурация в git.

Настраиваю utf-8 в выходе сообщений git

```
grkurushin@vbox:~$ git config --global user.email 1132246755@pfur.ru
grkurushin@vbox:~$ git config --global core.quotepath false
grkurushin@vbox:~$
```

Рис. 4.3: Настройка utf-8.

Задаю имя начальной ветки, которую буду называть master

```
grkurushin@vbox:~$ git config --global init.defaultBranch master
grkurushin@vbox:~$
```

Рис. 4.4: Имя начальной ветки.

А также ввожу autocrlf и safecrlf

```
grkurushin@vbox:~$ git config --global core.autocrlf input
grkurushin@vbox:~$ git config --global core.safecrlf warn
grkurushin@vbox:~$
```

Рис. 4.5: Ввод команд autocrlf и safecrlf.

4.3 Создание SSH ключа

Для последующей идентификации пользователя на сервере репозитория генерирую пару ключей (приватный и открытый).

```
grkurushin@vbox:~$ ssh-keygen -C "Georgy Kurushin <1132246755@pfur.ru>"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/grkurushin/.ssh/id_ed25519):
Created directory '/home/grkurushin/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/grkurushin/.ssh/id_ed25519
Your public key has been saved in /home/grkurushin/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:FijzBqdLfubnK/AA1q8V3fjodHtaAz74MiyPicpLxbQ Georgy Kurushin <1132246755@pfur.ru>
The key's randomart image is:
+--[ED25519 256]--+
|
|..+
|..+
|..+
|..+
|..+
|..+
|..+
|..+
|..+
|..+
+-----[SHA256]-----+
grkurushin@vbox:~$
```

Рис. 4.6: Генерация ключей.

Чтобы скопировать из локальной консоли ключ в буфер обмена, устанавливаю команду xclip

```
grkurushin@vbox:~$ xclip
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.git11c6a61.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...
```

Рис. 4.7: Скачивание команды xclip.

Теперь воспользуюсь командой xclip

```
grkurushin@vbox:~$ ssh-keygen -C "Georgy Kurushin <1132246755@pfur.ru>"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/grkurushin/.ssh/id_ed25519):
/home/grkurushin/.ssh/id_ed25519 already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/grkurushin/.ssh/id_ed25519
Your public key has been saved in /home/grkurushin/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:DA9Sp/uc6hOBmL15gEcN5AY/N3j1IivKanMwtJOUCq4 Georgy Kurushin <1132246755@pfur.ru>
The key's randomart image is:
+--[ED25519 256]--+
|.oo. o .|
| o..oo.+|
| +xB+= .|
|. +B=+.B|
|o =.++..S|
|. + *o oo .|
|o. o. .+|
|Eo o ..|
|o + .o.|
+-----[SHA256]-----+
grkurushin@vbox:~$ cat ~/.ssh/id_ed25519.pub | xclip -sel clip
grkurushin@vbox:~$
```

Рис. 4.8: Копирование ключа.

Вставляю ключ в появившееся на сайте поле, указываю его имя.

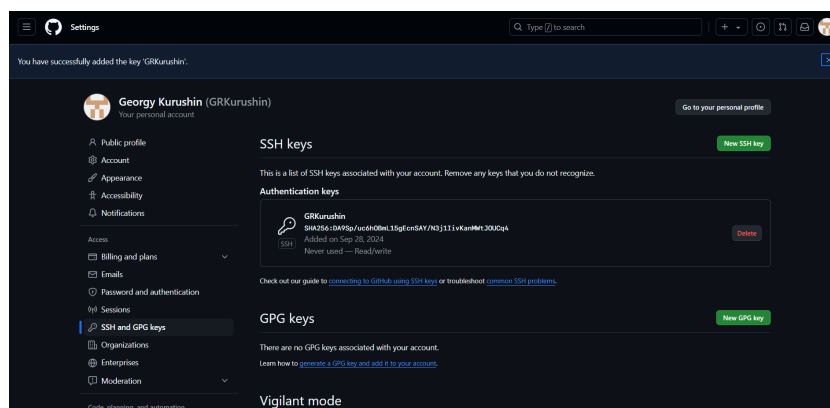


Рис. 4.9: Вставка ключа.

4.4 Создание рабочего пространства, репозитория курса на основе шаблона

Открываю терминал и создаю репозиторий для предмета «Архитектура компьютера»

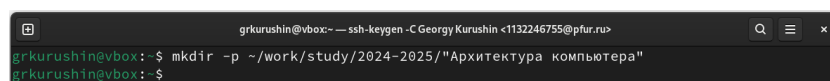


Рис. 4.10: Создание терминала для предмета «Архитектура компьютера».

4.5 Создание репозитория курса на основе шаблона

Захожу на страницу репозитория с шаблоном курса, выбираю его в качестве своего нового

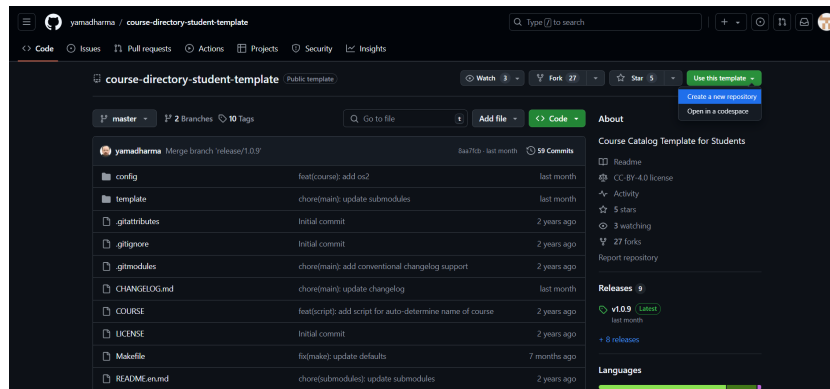


Рис. 4.11: Выбор шаблона

Далее создаю его, задав ему имя

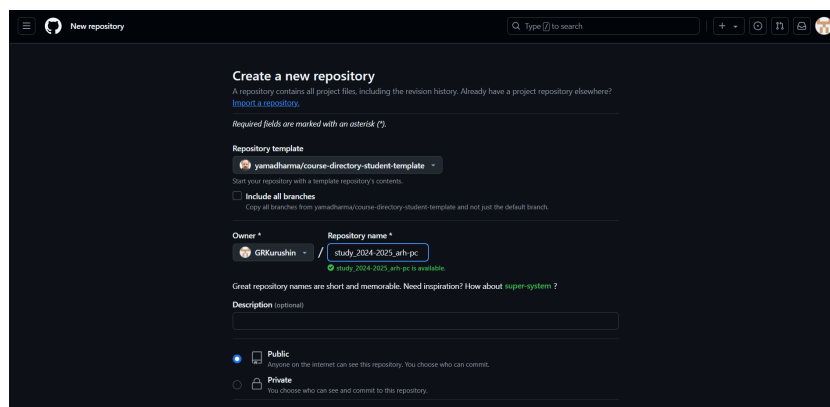


Рис. 4.12: Создание репозитория.

Открываю терминал и перехожу в каталог курса

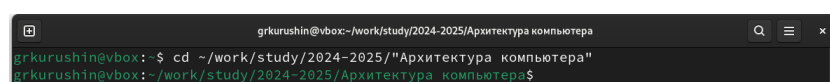


Рис. 4.13: Переход в каталог курса

Клонирую созданный репозиторий

```
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера$ git clone --recursive git@github.com:GRKurushin/study_2024-2025_arh-pc.git arch-pc
Клонирование в «arch-pc»...
remote: Enumerating objects: 33, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (32/32), done.
remote: Total 33 (delta 1), reused 18 (delta 0), pack-reused 0 (from 0)
Получение объектов: 100% (33/33), 18.81 КиБ | 437.00 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 «/home/grkurushin/work/study/2024-2025/Архитектура компьютера/arch-pc/template/presentation»...
remote: Enumerating objects: 111, done.
remote: Counting objects: 100% (111/111), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 111 (delta 42), reused 100 (delta 31), pack-reused 0 (from 0)
Получение объектов: 100% (111/111), 102.17 КиБ | 316.00 КиБ/с, готово.
Определение изменений: 100% (42/42), готово.
Клонирование в «/home/grkurushin/work/study/2024-2025/Архитектура компьютера/arch-pc/template/report»...
remote: Enumerating objects: 142, done.
remote: Counting objects: 100% (142/142), done.
remote: Compressing objects: 100% (97/97), done.
remote: Total 142 (delta 60), reused 121 (delta 39), pack-reused 0 (from 0)
Получение объектов: 100% (142/142), 341.09 КиБ | 156.00 КиБ/с, готово.
Определение изменений: 100% (60/60), готово.
Submodule path 'template/presentation': checked out 'c9b2712b4b2d431ad5086c9c72a02bd2fca1d4a6'
Submodule path 'template/report': checked out 'c26e22effe7b3e0495707d82ef561ab185f5c748'
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера$
```

Рис. 4.14: Клонирование репозитория.

4.6 Настройка каталога курса

Перехожу в каталог курса и удаляю лишние файлы

```
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cd arch-pc
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ rm package.json
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$
```

Рис. 4.15: Удаление лишних файлов в каталоге курса.

Создаю необходимые каталоги, отправляю файлы на сервер

```
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ echo arch-pc > COURSE
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ make prepare
make: «prepare» не требует обновления.
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git add .
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git commit -am 'feat(main): make course structure'
[master a75a0aa] feat(main): make course structure
223 files changed, 53681 insertions(+), 14 deletions(-)
create mode 100644 labs/README.md
create mode 100644 labs/README.ru.md
create mode 100644 labs/lab01/presentation/.projectile
create mode 100644 labs/lab01/presentation/.texlabroot
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
```

Рис. 4.16: Создание каталогов и их отправка на сервер.

В локальном репозитории проверяю результат выполненной работы

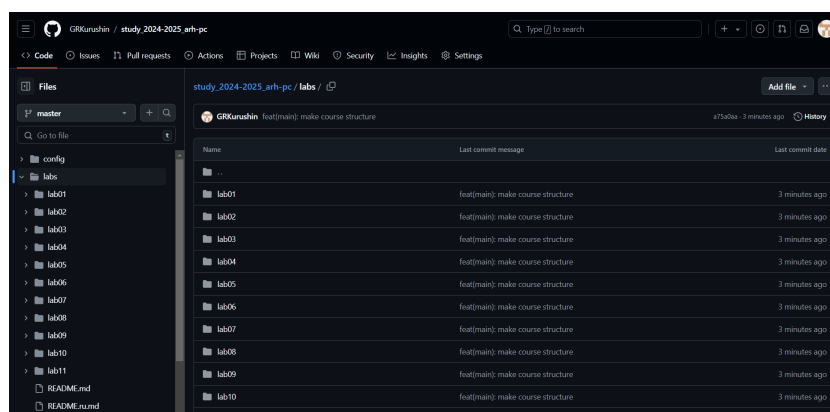


Рис. 4.17: Проверка выполненной работы.

4.7 Выполнение заданий для самостоятельной работы

Создаю отчет по выполнению второй лабораторной работы в соответствующем каталоге. С помощью команды `ls` проверяю, создан ли файл

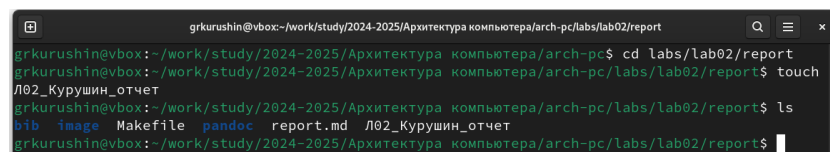


Рис. 4.18: Создание отчета о выполнении работы.

Для выполнения второго задания проверяю местонахождение своих лабораторных работ

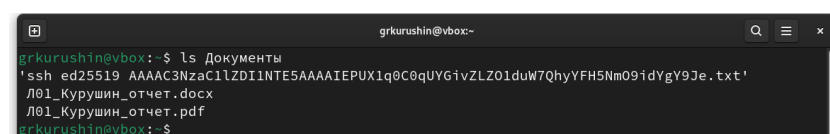
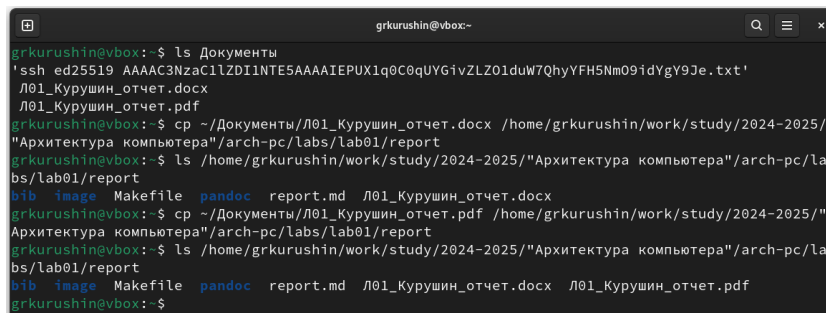


Рис. 4.19: Местонахождение лабораторных работ.

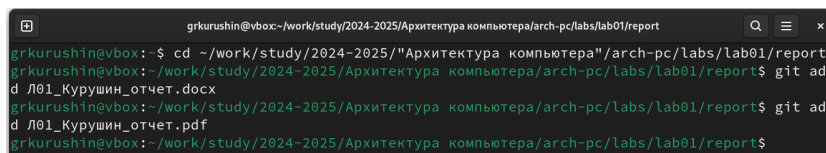
Копирую лабораторную работу с помощью утилиты `sr`, проверяю местонахождение файлов с помощью команды `ls`

A terminal window titled 'grkurushin@vbox:~' showing a series of commands to copy files. The user lists files in the 'Документы' directory, then copies 'Л01_Курушин_отчет.docx' and 'Л01_Курушин_отчет.pdf' to a specific directory path. Finally, they run 'ls' to verify the files are in the correct location.

```
grkurushin@vbox:~$ ls Документы
'ssh ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIEPUX1q0C0qUYGivZLZ01duW7QhyYFH5Nm09idVgY9Je.txt'
Л01_Курушин_отчет.docx
Л01_Курушин_отчет.pdf
grkurushin@vbox:~$ cp ~/Документы/Л01_Курушин_отчет.docx /home/grkurushin/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab01/report
grkurushin@vbox:~$ ls /home/grkurushin/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab01/report
b1b image Makefile pandoc report.md Л01_Курушин_отчет.docx
grkurushin@vbox:~$ cp ~/Документы/Л01_Курушин_отчет.pdf /home/grkurushin/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab01/report
grkurushin@vbox:~$ ls /home/grkurushin/work/study/2024-2025/"Архитектура компьютера"/arch-pc/labs/lab01/report
b1b image Makefile pandoc report.md Л01_Курушин_отчет.docx Л01_Курушин_отчет.pdf
grkurushin@vbox:~$
```

Рис. 4.20: Копирование отчета по лабораторной работе в нужный каталог.

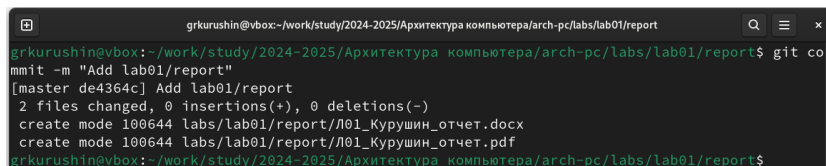
Для того чтобы загрузить эти файлы на GitHub, в первую очередь я использую команду `git add`. Так добавленные мной файлы станут отслеживаемыми

A terminal window showing the user navigating to the directory where the files were copied and running the 'git add' command to stage the files for commit.

```
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report$ git add
Л01_Курушин_отчет.docx
Л01_Курушин_отчет.pdf
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report$
```

Рис. 4.21: Добавление файлов с помощью команды `git add`.

Теперь осуществляю полноценный перенос файлов с помощью команды `git commit -m "..."`

A terminal window showing the user running 'git commit -m "Add lab01/report"'. The output shows that two files were added to the repository.

```
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report$ git commit -m "Add lab01/report"
[master de4364c] Add lab01/report
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 labs/lab01/report/Л01_Курушин_отчет.docx
create mode 100644 labs/lab01/report/Л01_Курушин_отчет.pdf
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report$
```

Рис. 4.22: Поручаю консоли совершить изменения.

Использую команды: `git status` и `git push`, чтобы опубликовать свои локальные КОММИТЫ


```
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report$ git status
Текущая ветка: master
Ваша ветка опережает «origin/master» на 1 коммит.
(используйте «git push», чтобы опубликовать ваши локальные коммиты)

Неотслеживаемые файлы:
(используйте «git add <файл>...», чтобы добавить в то, что будет включено в коммит)
.../..lab02/report/002_Курушин_отчет

индекс пуст, но есть неотслеживаемые файлы
(используйте «git add», чтобы проиндексировать их)
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01/report$ cd ..
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab01$ cd ..
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs$ cd ..
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ git push
Перечисление объектов: 11, готово.
Подсчет объектов: 100% (11/11), готово.
При сжатии изменений используется до 4 потоков
Сжатие объектов: 100% (7/7), готово.
Запись объектов: 100% (7/7), 3.04 МБ | 539.00 КиБ/с, готово.
Total 7 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:GRKurushin/study_2024-2025_arh-pc.git
 a75a0aa..de4364c master -> master
grkurushin@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc$
```

Рис. 4.23: Команды git status и git push для завершения копирования.

Перехожу в каталоги на GitHub, чтобы убедиться в том, что файлы находятся в нужных репозиториях

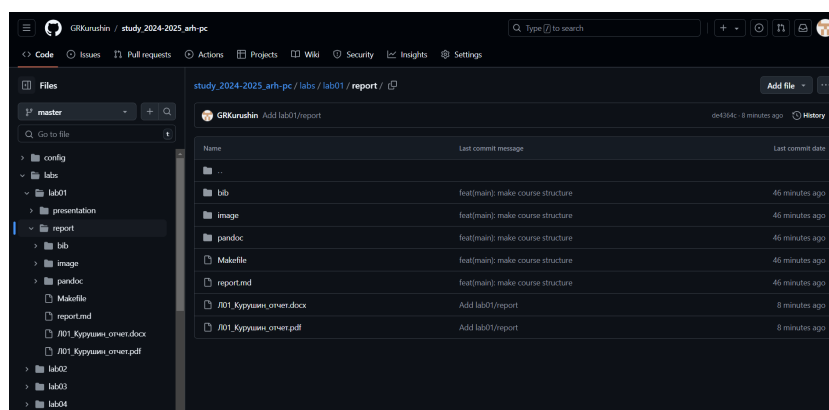


Рис. 4.24: Проверка проделанных операций.

5 Выводы

В заключение хочется отметить, что данная лабораторная работа позволила мне научиться работать с системой Git. Я практиковал свои навыки в работе с командной строкой, теперь уже связывая выполнимое с директориями GitHub.

Список литературы

Архитектура ЭВМ

30 команд Git, необходимых для освоения интерфейса командной строки Git

Система контроля версий: определение, функции, популярные решения