

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: Головина М.И.

Группа: НММбд-02-24

МОСКВА

2024 г.

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
5	Выводы	24
	Список литературы	25

Список иллюстраций

4.1 Стартовая страница сайта github.com.....	11
4.2.1 Ввод команды имени владельца.....	11
4.2.2 Ввод команды почты владельца.....	12
4.2.3 Ввод команды настройки utf-8 в выводе сообщений git.....	12
4.2.4 Имя начальной ветки (master)	13
4.2.5 Параметр autocrlf.....	13
4.2.6 Параметр safecrlf.....	14
4.3.1 Ввод команды для генерации ключей.....	14
4.3.2 Копирование ключа в буфер обмена.....	15
4.3.3 Ключ.....	15
4.4 Создание каталога «Архитектура компьютера».....	16
4.5.1 Создание репозитория.....	16
4.5.2 Переход в каталог.....	17
4.5.3 Клонирование репозитория.....	17
4.6.1 Каталог курса.....	18
4.6.2 Ввод команды для удаления.....	18
4.6.3 Ввод команды для создания каталогов.....	19
4.6.4 Отправка данных на сервер.....	19
4.6.5 Отправка данных на сервер.....	20
4.6.6 Отправка данных на сервер.....	20
4.6.7 Проверка правильности создания иерархии рабочего пространства	21
4.7.1 Создание каталога.....	21
4.7.2 Создание каталога для предыдущих лабораторных работ.....	22
4.7.3 Загрузка файлов на github.....	22
4.7.4 Проверка файлов на github.....	23

Список таблиц

3.1. Основные команды git.....	10
--------------------------------	----

1 Цель работы

Изучить идеологию и применение средств контроля версии. Приобрести практические навыки по работе с системой git.

2 Задание

1. Открыть сайт <https://github.com/>, создать учётную запись, заполнить основные данные.
2. Создать предварительную конфигурацию git. Открыть терминал и ввести необходимые команды, указав имя и email владельца репозитория.
3. Настроить utf-8 в выводе сообщений git.
4. Задать имя начальной ветки (master).
5. Настроить параметр autocrlf.
6. Настроить параметр safecrlf.
7. Сгенерировать пару ключей (приватный и открытый).
8. Скопировать из локальной консоли ключ в буфер обмена.
9. Внести ключ на сайте, указав имя ключа Title.
10. Создать каталог для предмета «Архитектура компьютера».
11. Создать репозиторий через web-интерфейс github.
12. Клонировали созданный репозиторий.
13. Перейти в каталог курса.
14. Удалить лишнее файлы.
15. Создать необходимые каталоги.
16. Отправить необходимые файлы на сервер.
17. Проверить правильность создания иерархии рабочего пространства в локальном репозитории и на странице github.

Задание для самостоятельной работы

1. Создать каталог для внесения отчета по выполнению лабораторной работы в (labs>lab02>report).
2. Создать каталог для внесения предыдущих лабораторных работ.
3. Скопировать отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства.
4. Загрузить и проверить файлы на github.

3 Теоретическое введение

3.1 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 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.

3.2 Система контроля версий Git

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.

3.3 Основные команды git

Наиболее часто используемые команды `git` представлены в таблице 3.1.

Таблица 3.1. Основные команды `git`

Команда	Описание
<code>git init</code>	создание основного дерева репозитория
<code>git pull</code>	получение обновлений (изменений) текущего дерева из центрального репозитория
<code>git push</code>	отправка всех произведённых изменений локального дерева в центральный репозиторий
<code>git status</code>	просмотр списка изменённых файлов в текущей директории
<code>git diff</code>	просмотр текущих изменения

git add .	добавить все изменённые и/или созданные файлы и/или каталоги
git add имена_файлов	имена_файлов добавить конкретные изменённые и/или созданные файлы и/или каталоги
git rm имена_файлов	удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории)
git commit -am 'Описание коммита'	сохранить все добавленные изменения и все изменённые файлы
git checkout -b имя_ветки	создание новой ветки, базирующейся на текущей
git checkout имя_ветки	переключение на некоторую ветку (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)
git push origin имя_ветки	отправка изменений конкретной ветки в центральный репозиторий
git merge --no-ff имя_ветки	слияние ветки с текущим деревом
git branch -d имя_ветки	удаление локальной уже слитой с основным деревом ветки
git branch -D имя_ветки	принудительное удаление локальной ветки
git push origin :имя_ветки	удаление ветки с центрального репозитория

3.4 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 центральной 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

4.1 Настройка github

Открыли сайт <https://github.com/>, создали учётную запись и заполнили основные данные (рис. 4.1).

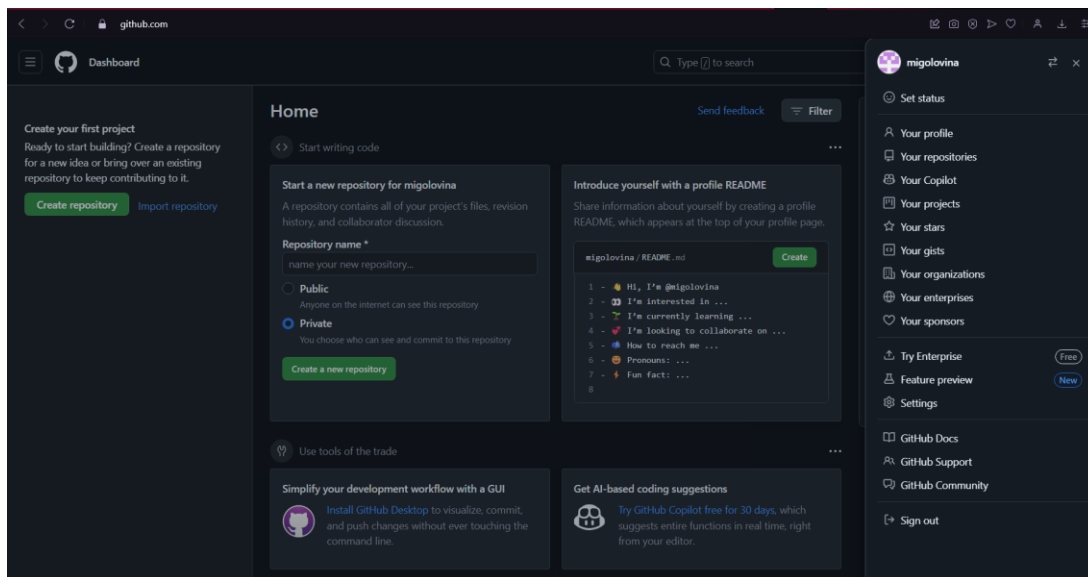


Рис. 4.1: Стартовая страница сайта github.com

4.2 Базовая настройка git

Создали предварительную конфигурацию git. Открыли терминал и ввели следующие команды, указав имя и email владельца репозитории (рис. 4.2.1-4.2.2)

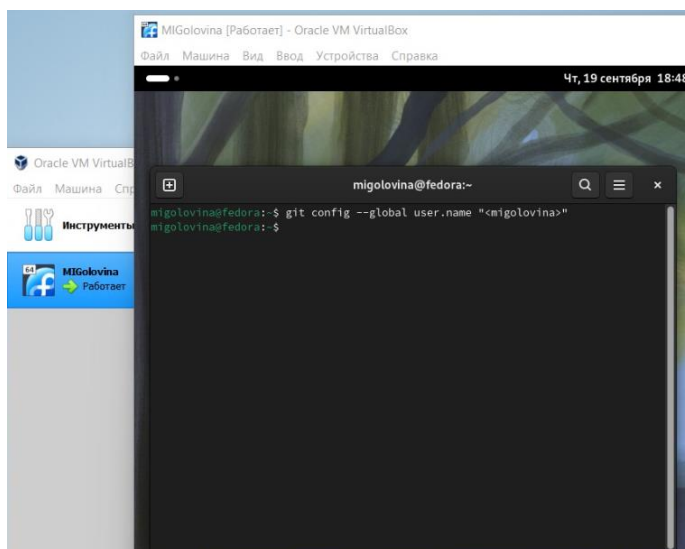


Рис. 4.2.1: Ввод команды имени владельца

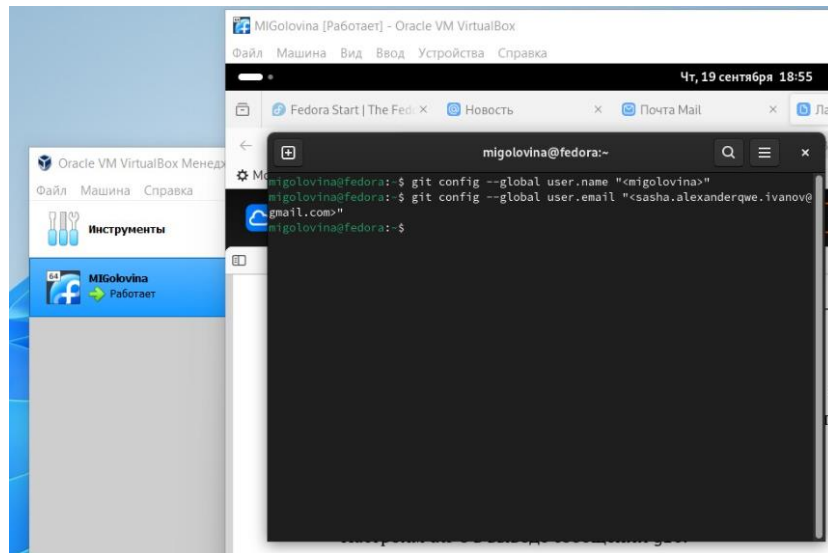


Рис. 4.2.2: Ввод команды почты владельца

Настроим utf-8 в выводе сообщений git (рис. 4.2.3).

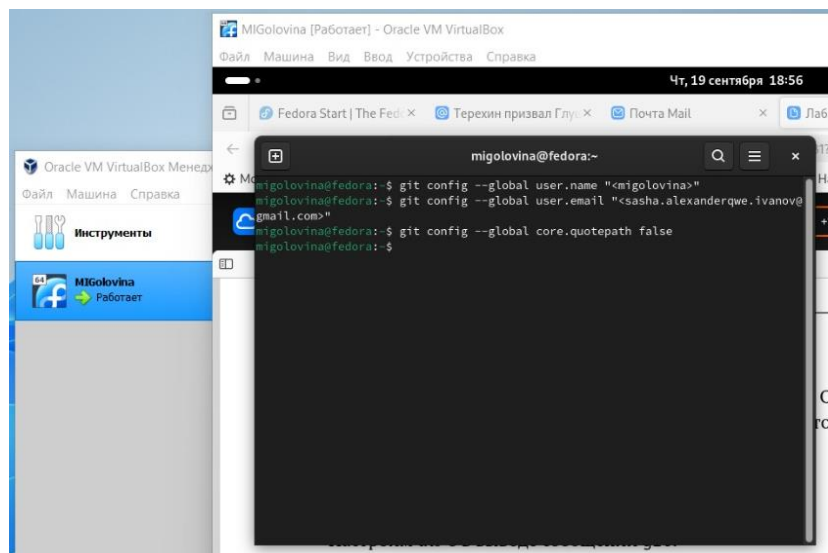


Рис. 4.2.3: Ввод команды настройки utf-8 в выводе сообщений git

Задали имя начальной ветки (рис. 4.2.4).

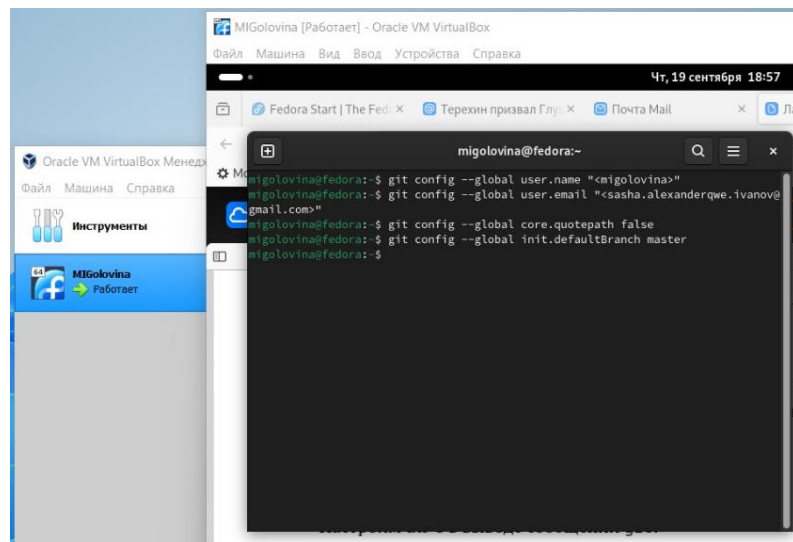


Рис. 4.2.4: Имя начальной ветки (master)

Настроили параметр autocrlf (рис. 4.2.5).

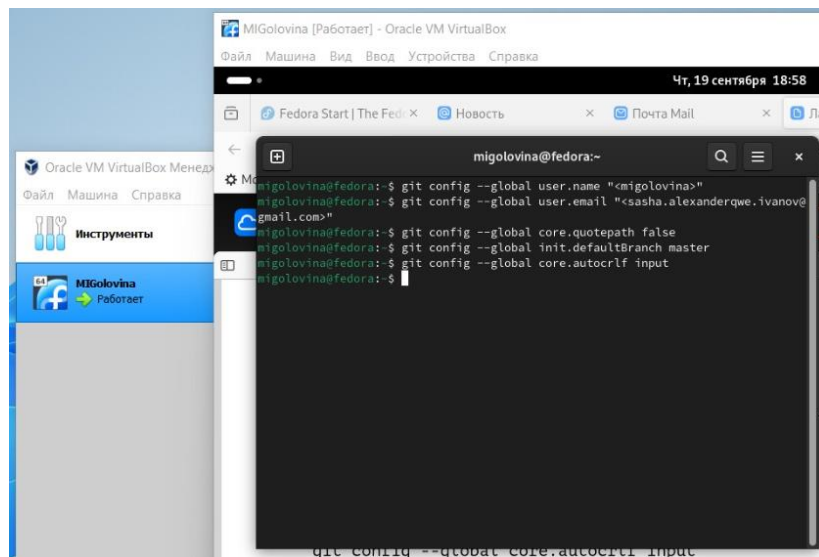


Рис. 4.2.5: Параметр autocrlf

Настроили параметр safecrlf (рис. 4.2.6)

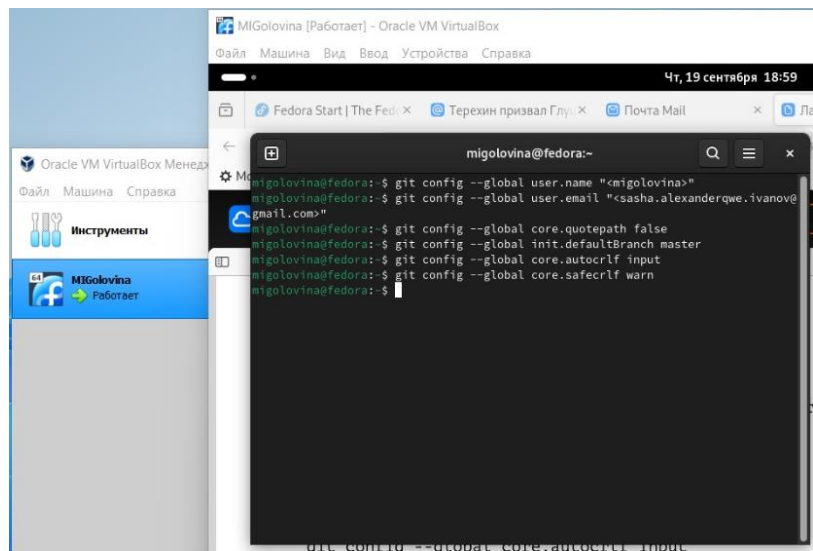


Рис. 4.2.6: Параметр safecrlf

4.3 Создание SSH ключа

Сгенерировали пару ключей (приватный и открытый) (рис. 4.3.1).

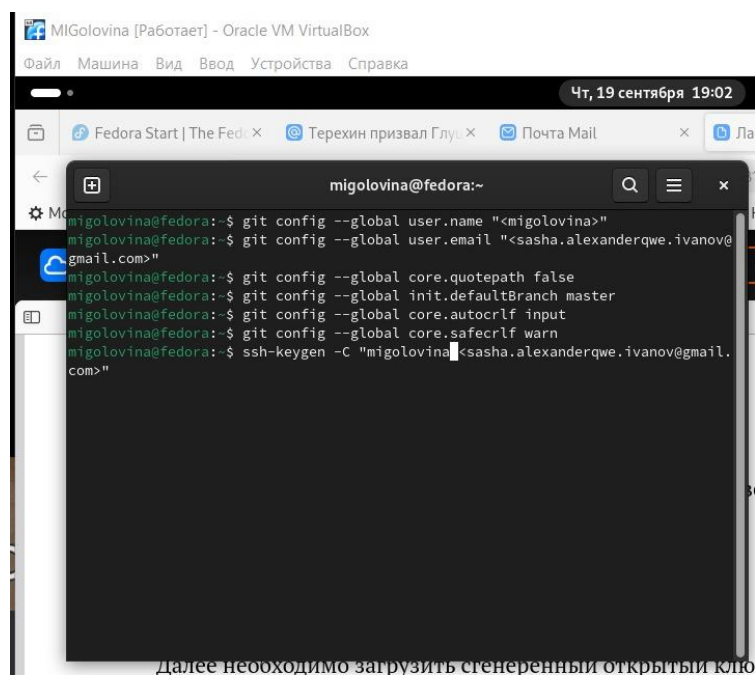


Рис. 4.3.1: Ввод команды для генерации ключей

Скопировали из локальной консоли ключ в буфер обмена (рис. 4.3.2).

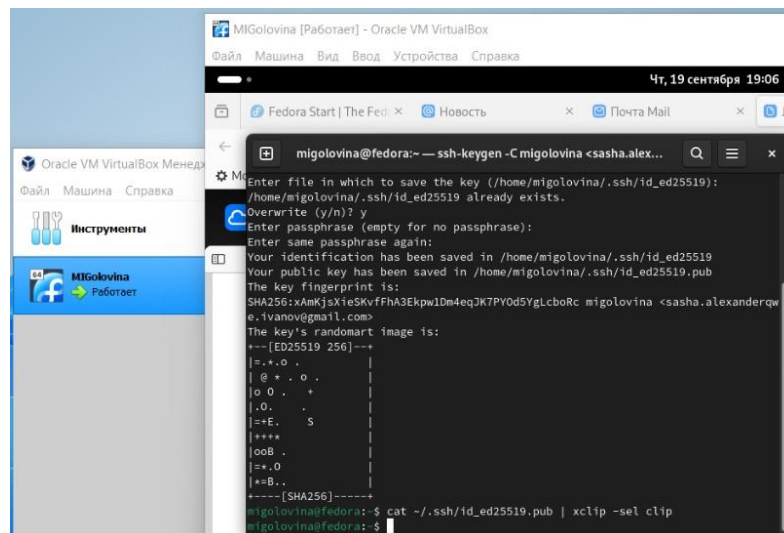


Рис. 4.3.2: Копирование ключа в буфер обмена

Внесли ключ на сайте, указали имя ключа Title (рис. 4.3.3).

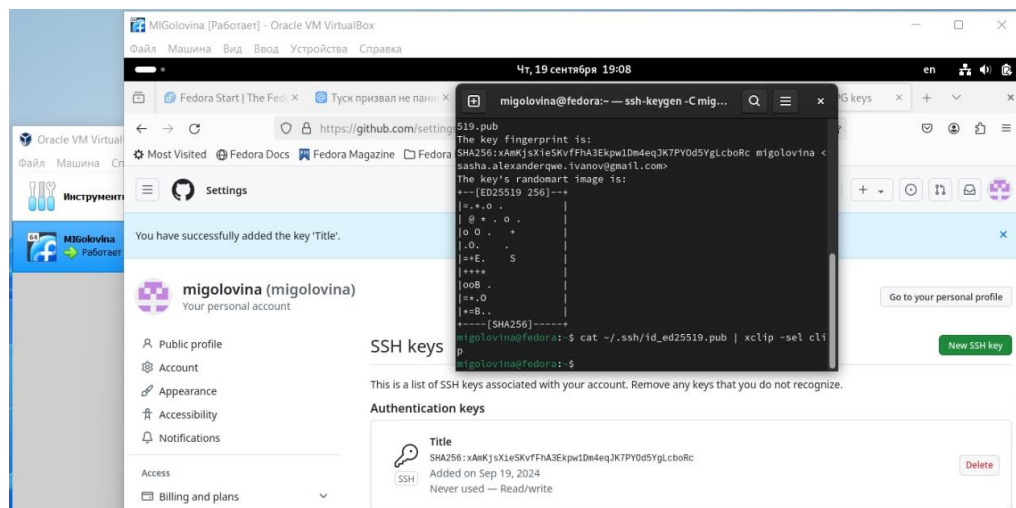


Рис. 4.3.3: Ключ

4.4 Сознание рабочего пространства и репозитория курса на основе шаблона

Создали каталог для предмета «Архитектура компьютера» (рис. 4.4)

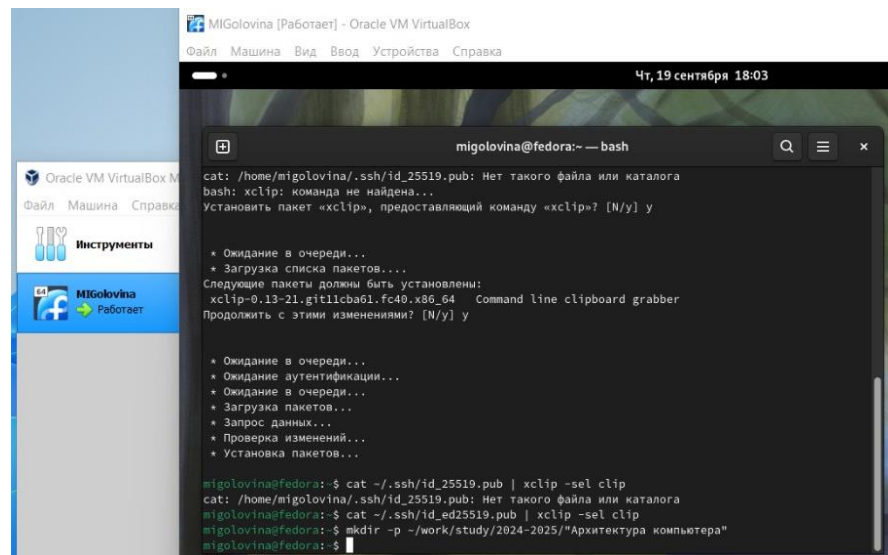


Рис. 4.4: Создание каталога «Архитектура компьютера»

4.5 Создание репозитория курса на основе шаблона

Создали репозиторий через web-интерфейс github (рис. 4.5.1).

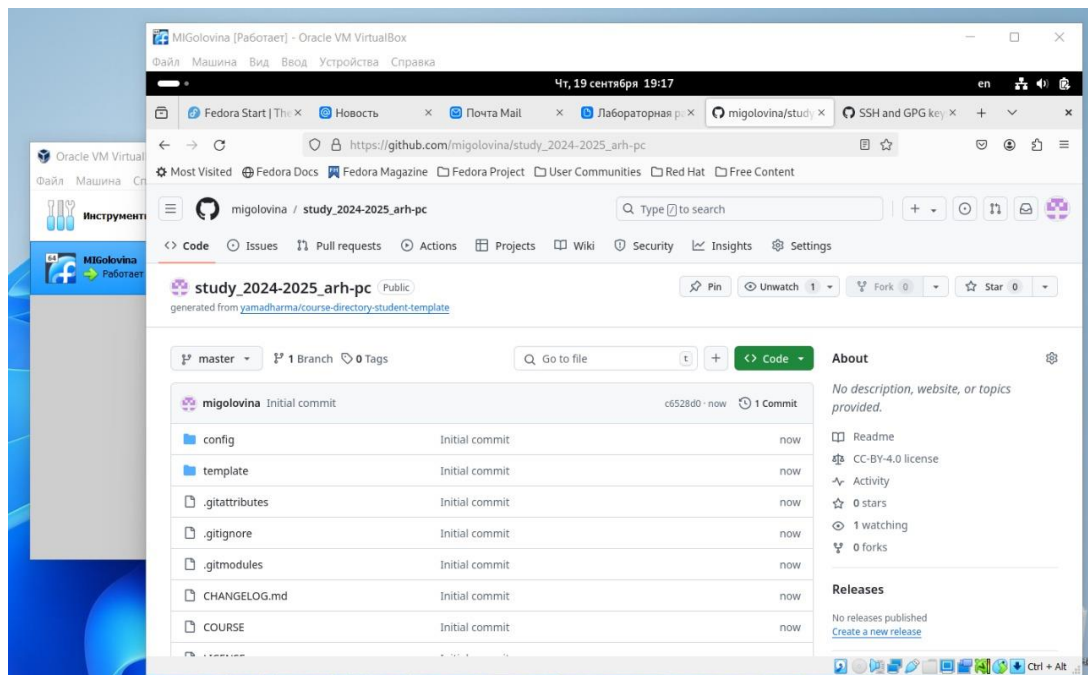


Рис. 4.5.1: Создание репозитория

Перешли в каталог курса (рис. 4.5.2).

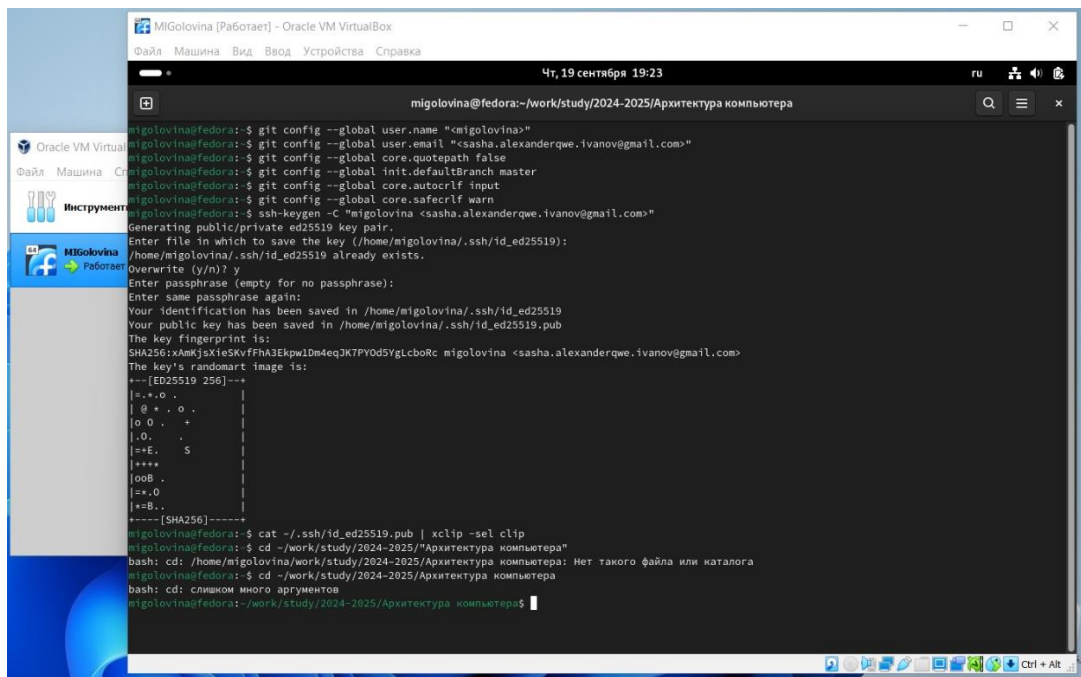


Рис. 4.5.2: Переход в каталог

Клонировали созданный репозиторий (рис. 4.5.3).

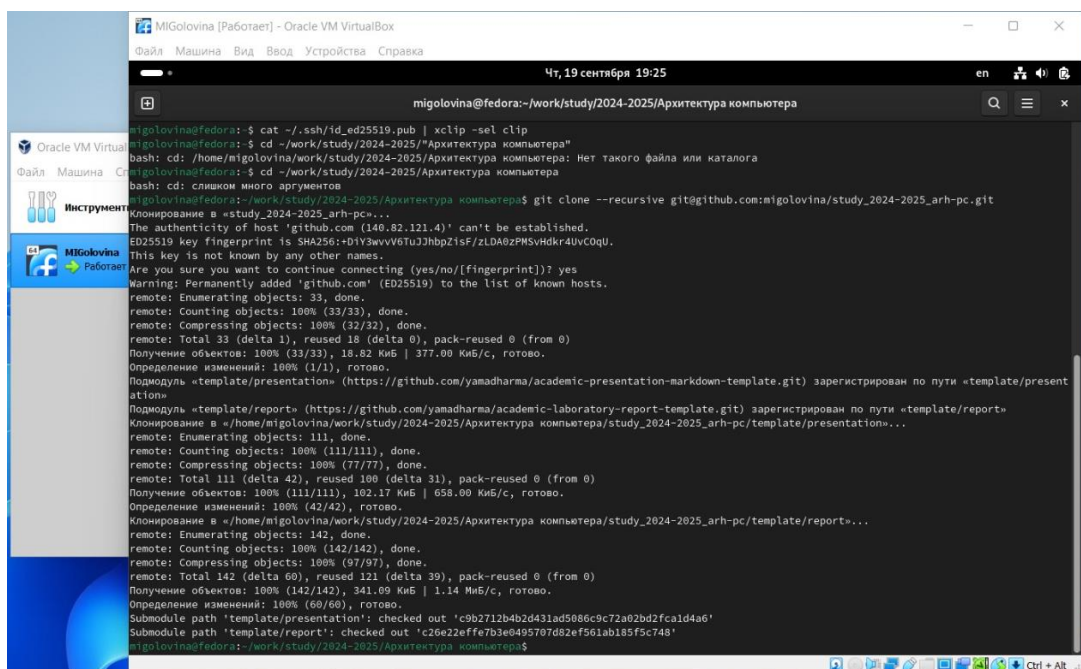


Рис. 4.5.3: Клонирование репозитория

4.6 Настройка каталога курса

Перешли в каталог курса (рис. 4.6.1).

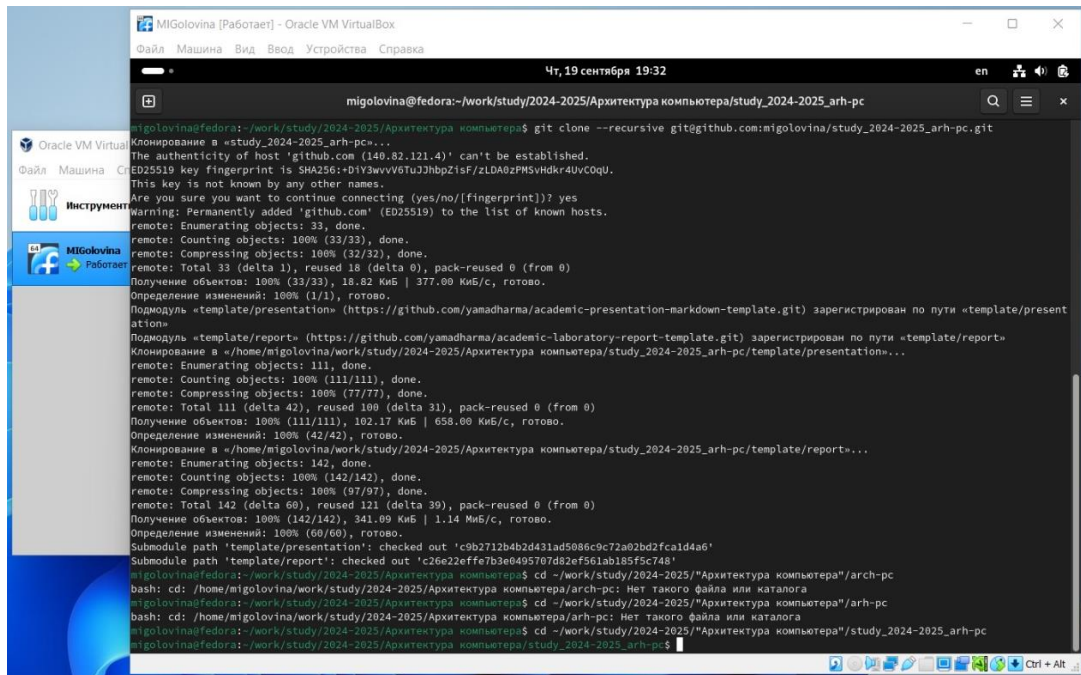


Рис. 4.6.1: Каталог курса

Удалили лишнее файлы (рис. 4.6.2).

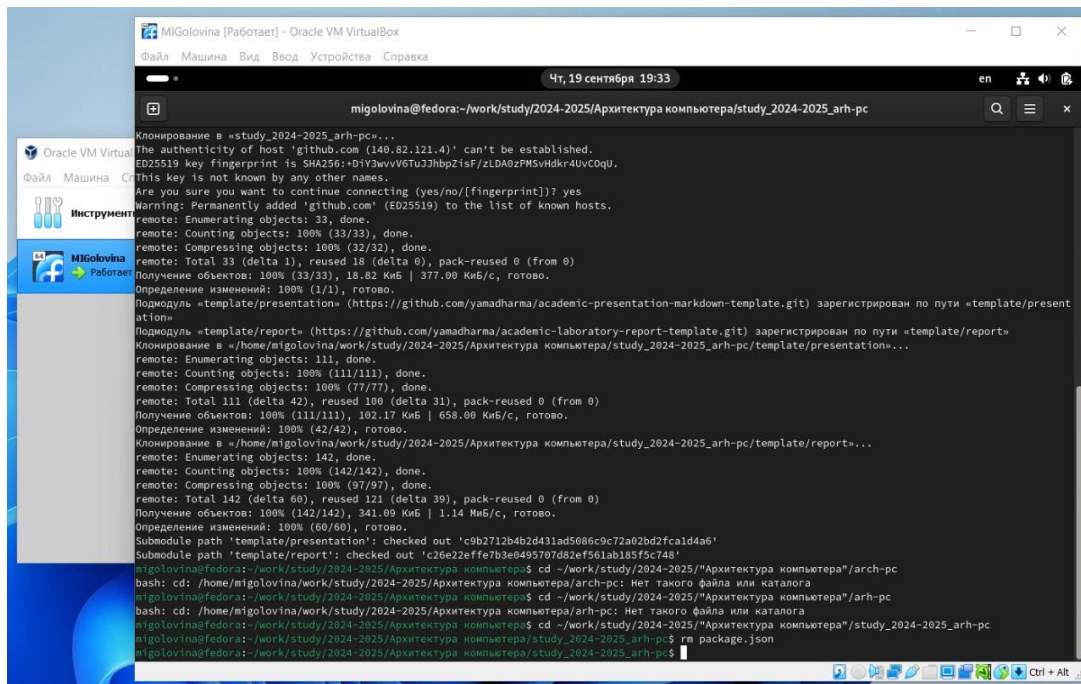
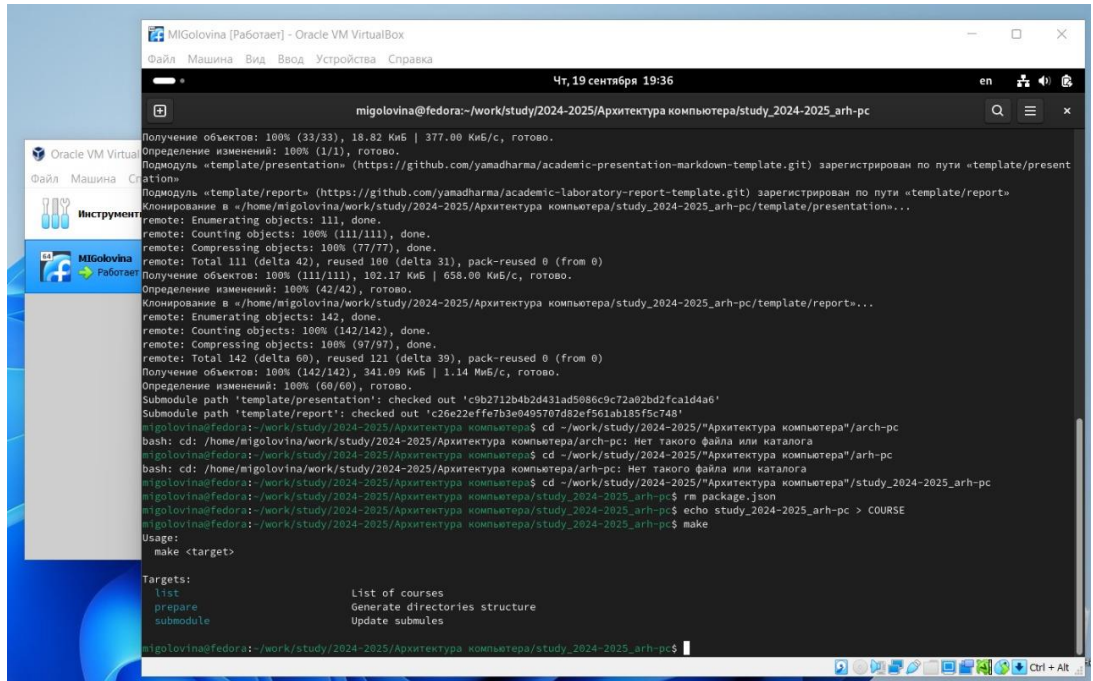


Рис. 4.6.2: Ввод команды для удаления

Создали необходимые каталоги (рис. 4.6.3).

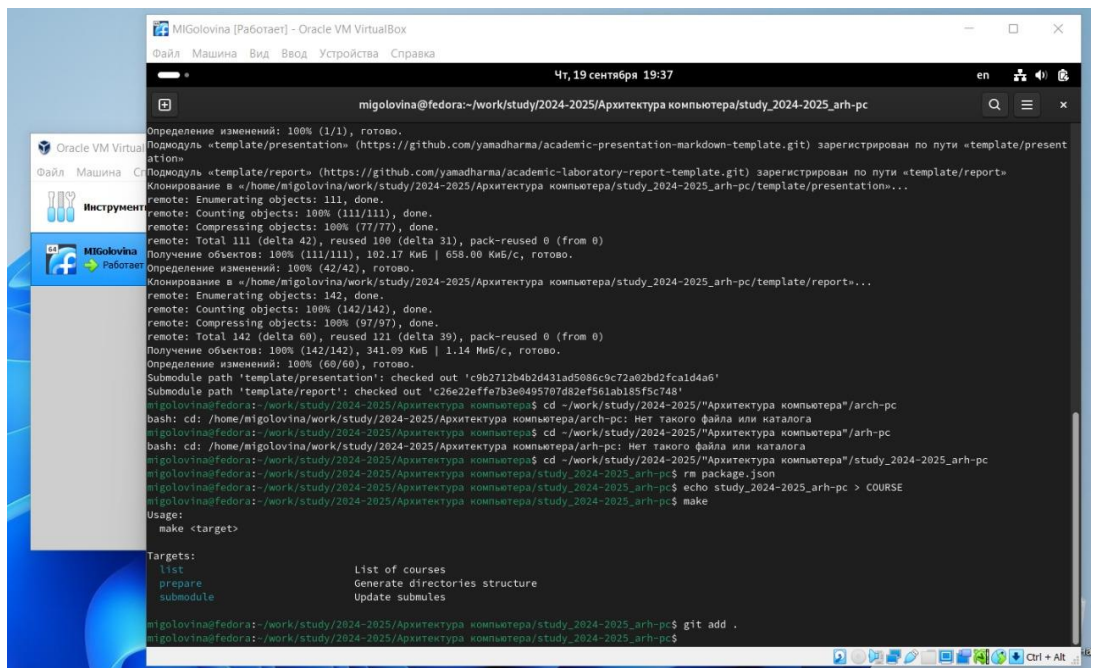


```
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc$ git add .
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc$ git commit -m "Initial commit"
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc$ git push
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc$ git clone https://github.com/yamadharma/academic-presentation-markdown-template.git
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc$ cd ~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template/
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template$ git init
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template$ git add .
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template$ git commit -m "Initial commit"
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template$ git push
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template$ git clone https://github.com/yamadharma/academic-laboratory-report-template.git
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template$ cd ~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template/report/
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template/report$ git init
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template/report$ git add .
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template/report$ git commit -m "Initial commit"
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template/report$ git push
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc$ make
Usage:
  make <target>

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

Рис. 4.6.3: Ввод команды для создания каталогов

Отправили необходимые файлы на сервер (рис. 4.6.4-4.6.6).



```
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc$ git add .
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc$ git commit -m "Initial commit"
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc$ git push
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc$ git clone https://github.com/yamadharma/academic-presentation-markdown-template.git
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc$ cd ~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template/
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template$ git init
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template$ git add .
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template$ git commit -m "Initial commit"
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template$ git push
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template$ git clone https://github.com/yamadharma/academic-laboratory-report-template.git
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template$ cd ~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template/report/
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template/report$ git init
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template/report$ git add .
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template/report$ git commit -m "Initial commit"
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc/template/report$ git push
migolovina@fedora:~/work/study/2024-2025/Архитектура компьютера/study_2024-2025_arh-pc$ make
Usage:
  make <target>

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

Рис. 4.6.4: Отправка данных на сервер

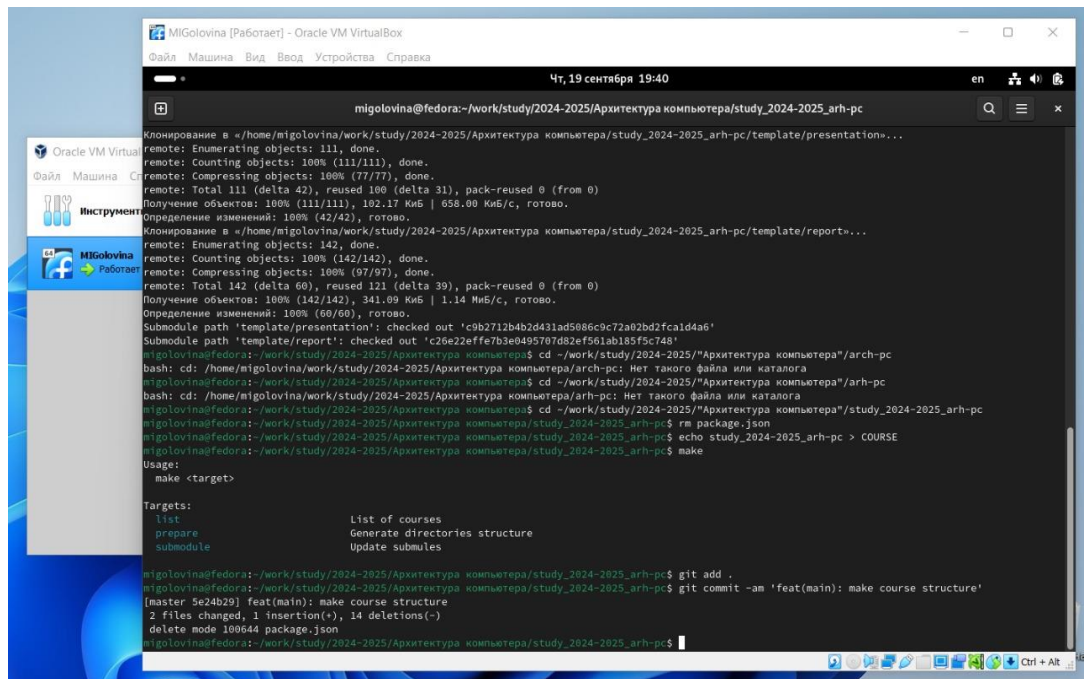


Рис. 4.6.5: Отправка данных на сервер

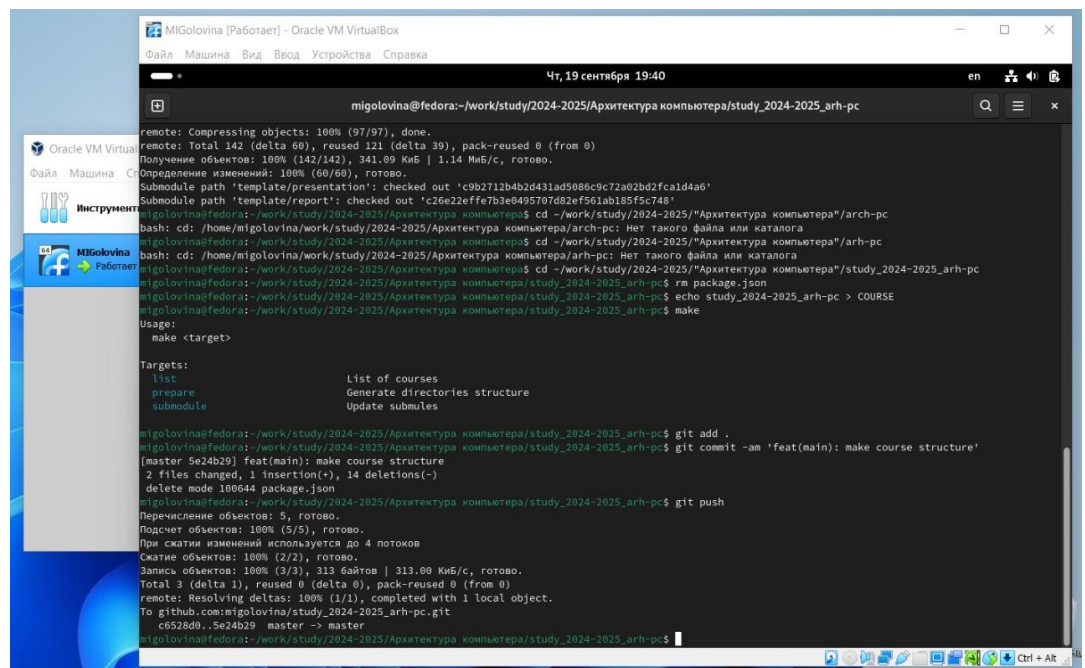


Рис. 4.6.6: Отправка данных на сервер

Проверили правильность создания иерархии рабочего пространства в локальном репозитории и на странице github (рис. 4.6.7).

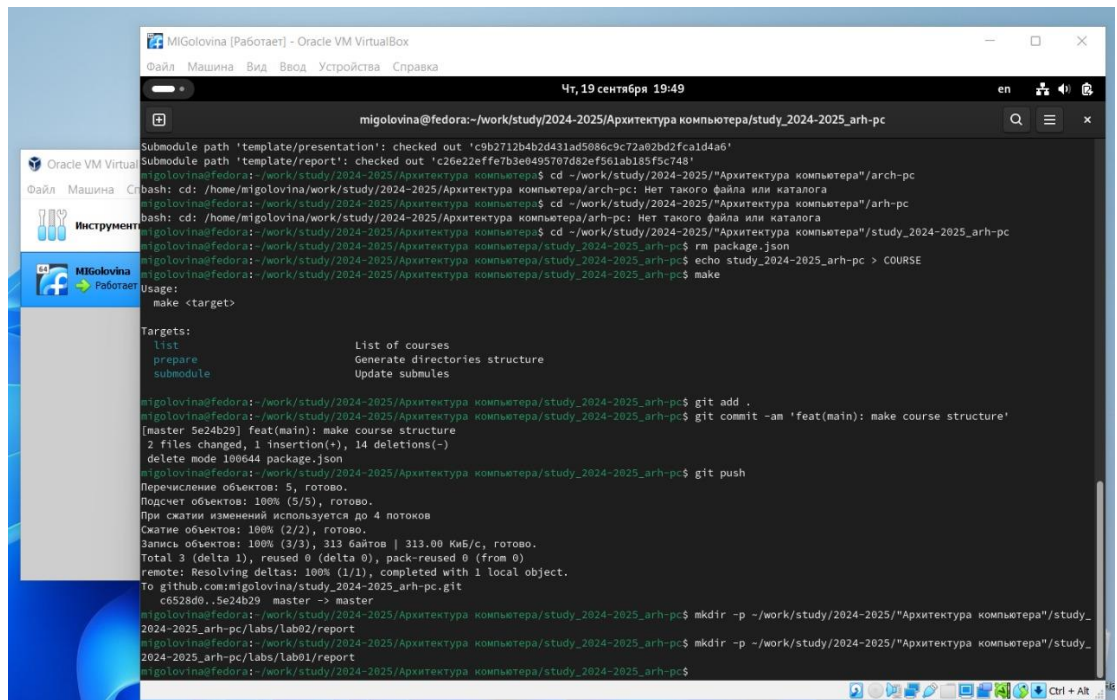


Рис. 4.7.2: Создание каталога для предыдущих лабораторных работ

Скопировали отчеты по выполнению предыдущих лабораторных работ в соответствующие каталоги созданного рабочего пространства.

Загрузили файлы на github (рис. 4.7.3-4.7.4).

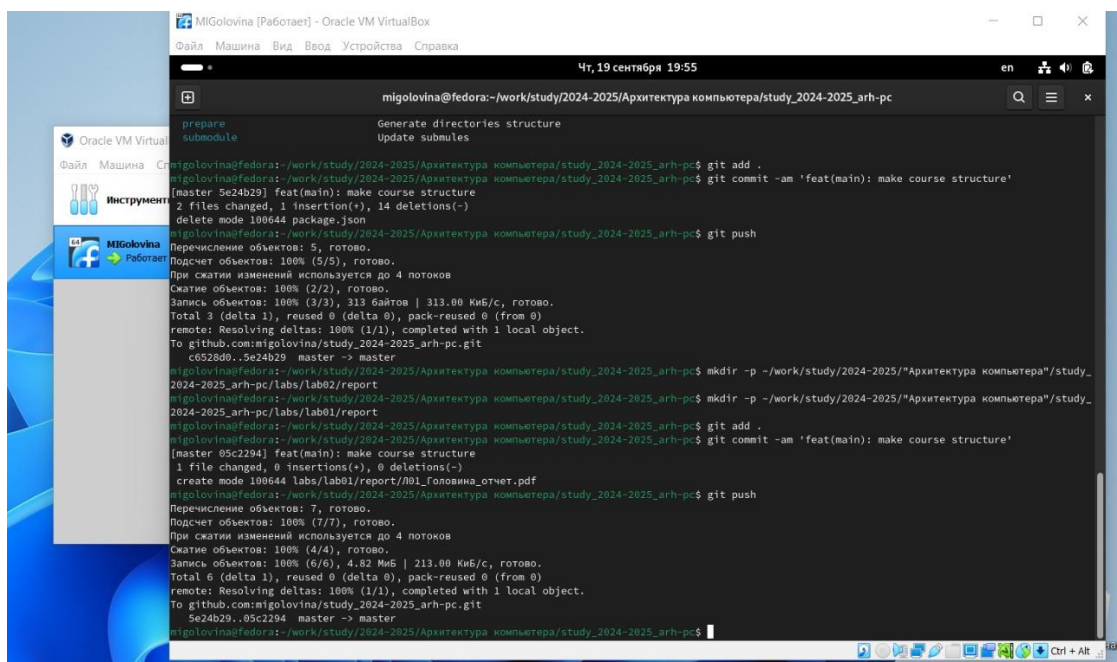


Рис. 4.7.3: Загрузка файлов на github

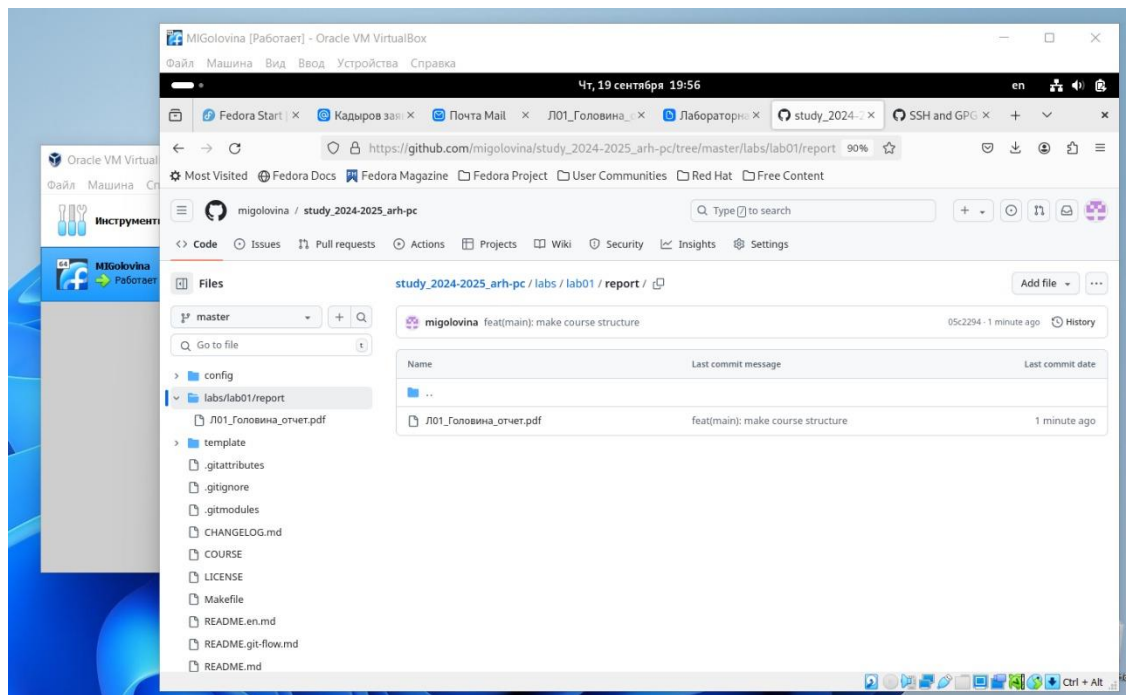


Рис. 4.7.4: Проверка файлов на github

5 Выводы

Изучили идеологию и применение средств контроля версии. Приобрели практические навыки по работе с системой git.

Список литературы

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.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ-Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).

16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд.
— СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).