

# **Отчёт по лабораторной работе №3**

**Дисциплина: Архитектура Компьютера**

Егор Витальевич Кузьмин

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Выводы</b>	<b>18</b>
	<b>Список литературы</b>	<b>19</b>

## **Список иллюстраций**

# 1 Цель работы

Целью данной работы является приобретение практического опыта работы с системой git, изучение принципов и применения контроля версий.

## 2 Задание

- 0. Общее ознакомление с git.
- 1. Настройка GitHub.
- 2. Базовая настройка Git.
- 3. Создание SSH-ключа.
- 4. Создание рабочего пространства.
- 5. Создание репозитория курса на основе шаблона.
- 6. Настройка каталога курса.
- 7. Выполнение задания 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 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

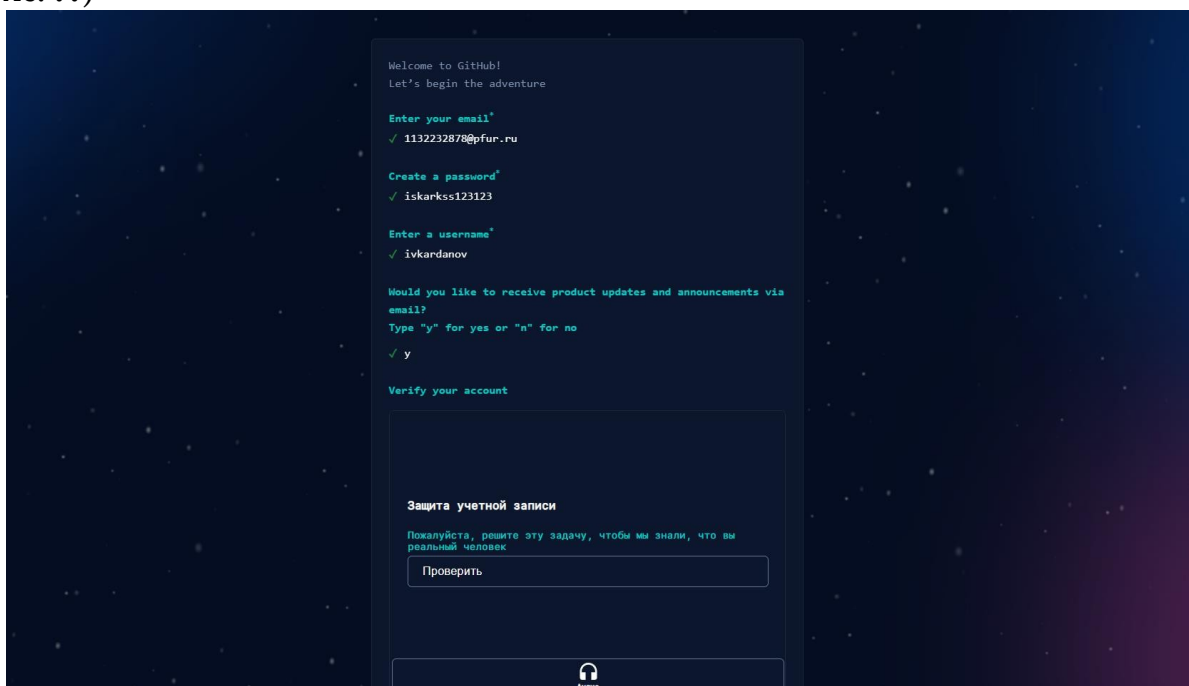
так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить изменения, сделанные разными участниками, вручную выбрать нужную версию, отменить изменения вовсе или 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. 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.

## 4 Выполнение лабораторной работы

### 1. Настройка GitHub

Создаю учетную запись на сайте GitHub, ввожу нужные данные учетной записи (рис. ??)



{#fig:1

width = 70%]

Можно персонализировать свой аккаунт (рис. ??)

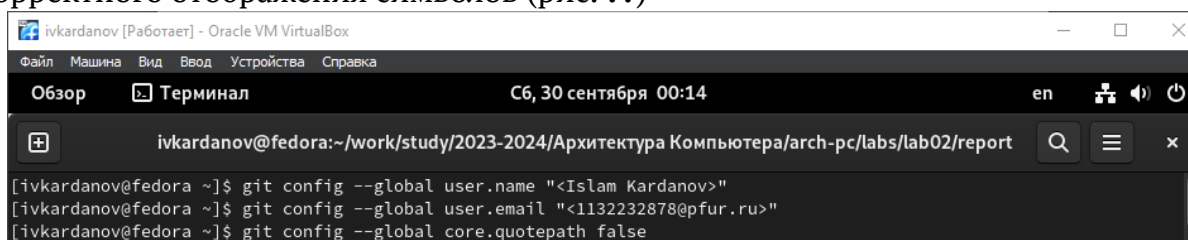
Персонализация учетной записи GitHub{#fig:2 width = 70%]

### 2. Базовая настройка Git

Открываю виртуальную машину, затем запускаю терминал, 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 этой команде свою электронную почту, настраиваю utf-8 в выводе сообщений git для корректного отображения символов (рис. ??)

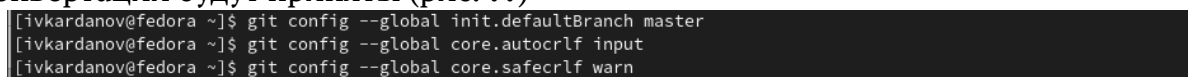


```
ivkardanov [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Обзор  Терминал  C6, 30 сентября 00:14  en  [иконки]
ivkardanov@fedora:~/work/study/2023-2024/Архитектура Компьютера/arch-pc/labs/lab02/report
[ivkardanov@fedora ~]$ git config --global user.name "<Islam Kardanov>"
[ivkardanov@fedora ~]$ git config --global user.email "<1132232878@pfur.ru>"
[ivkardanov@fedora ~]$ git config --global core.quotePath false
```

{#fig:3

width = 70%]

Задаю имя начальной ветке, а также параметры `autocrlf` и `safecrlf`, причем параметр `autocrlf` дополняем значением `input`, для конвертации символов разрыва строки в текстовых файлах (CRLF и LF) только при коммитах. Параметру `safecrlf` задаю значение `warn`, так Git будет проверять преобразование на обратимость, и при данном значении будет выведено только предупреждение, а необратимые конвертации будут приняты (рис. ??)



```
[ivkardanov@fedora ~]$ git config --global init.defaultBranch master
[ivkardanov@fedora ~]$ git config --global core.autocrlf input
[ivkardanov@fedora ~]$ git config --global core.safecrlf warn
```

{#fig:4

width = 70%]

### 3. Создание SSH-ключа

Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый). Для этого ввожу команду `ssh-keygen -C «Имя Фамилия, work@email»`, указывая своё имя и электронную почту. Ключ автоматически сохранится в каталоге `~/.ssh/`. (рис. ??)

Генерация SSH-ключа{#fig:5 width = 70%]

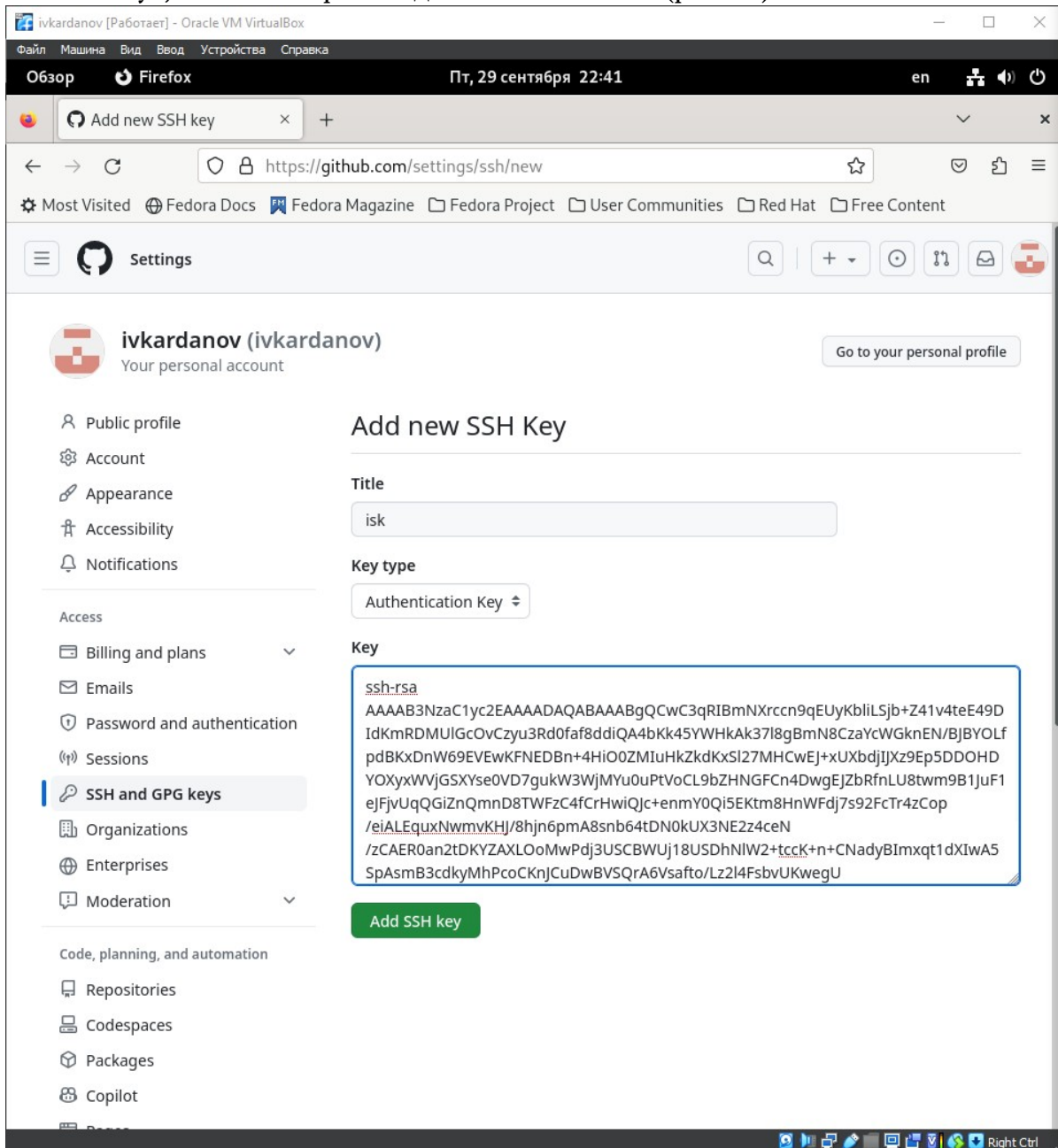
Устанавливаю утилиту `xclip`, позволяющую копировать любой текст через терминал. Использую команду `dnf install` с ключом `-u` от имени суперпользователя, ввожу в начале `sudo` (рис. ??) Установка утилиты `xclip`{#fig:6 width = 70%]

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

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

width = 70%]

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 (рис. ??)



width = 70%]

#### 4. Создание рабочего пространства

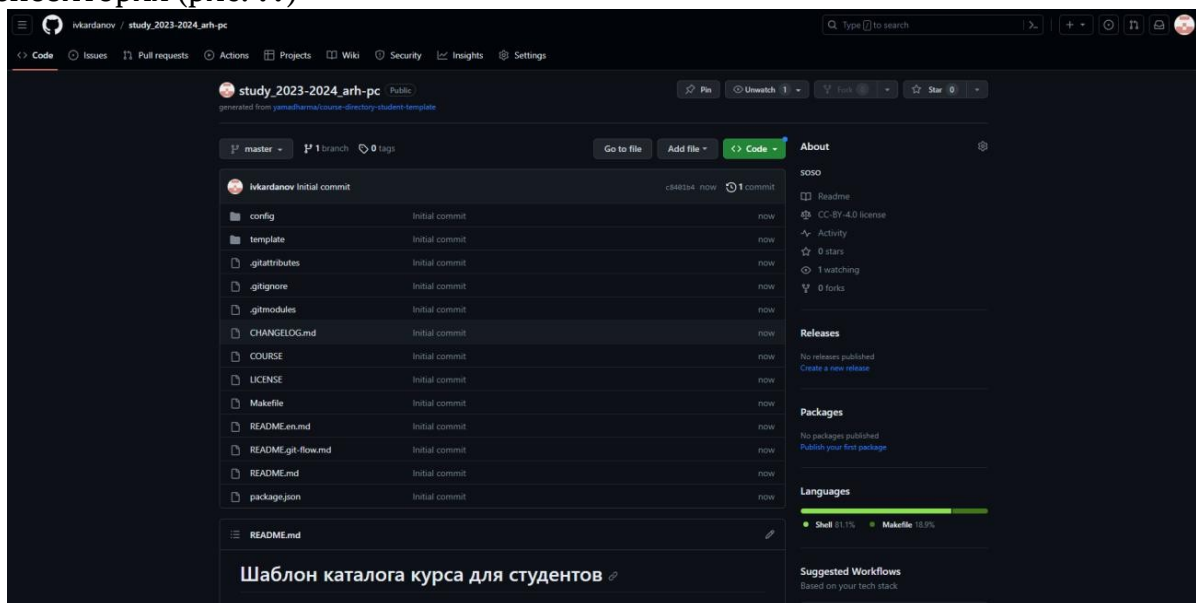
Закрываю браузер, открываю терминал. Создаю директорию, рабочее пространство с помощью утилиты mkdir. С помощью ключа -p создаю рекурсивно все директории после домашней ~/work/study/2023-2024/“Архитектура Компьютера”. Далее проверяю с помощью ls, действительно ли были созданы необходимые мне каталоги (рис. ??)

Создание рабочего пространства{#fig:9 width = 70%]

#### 5. Создание репозитория курса на основе шаблона

В браузере перехожу на страницу репозитория с шаблоном курса по адресу: <https://github.com/yamadharm/course-directory-student-template>.

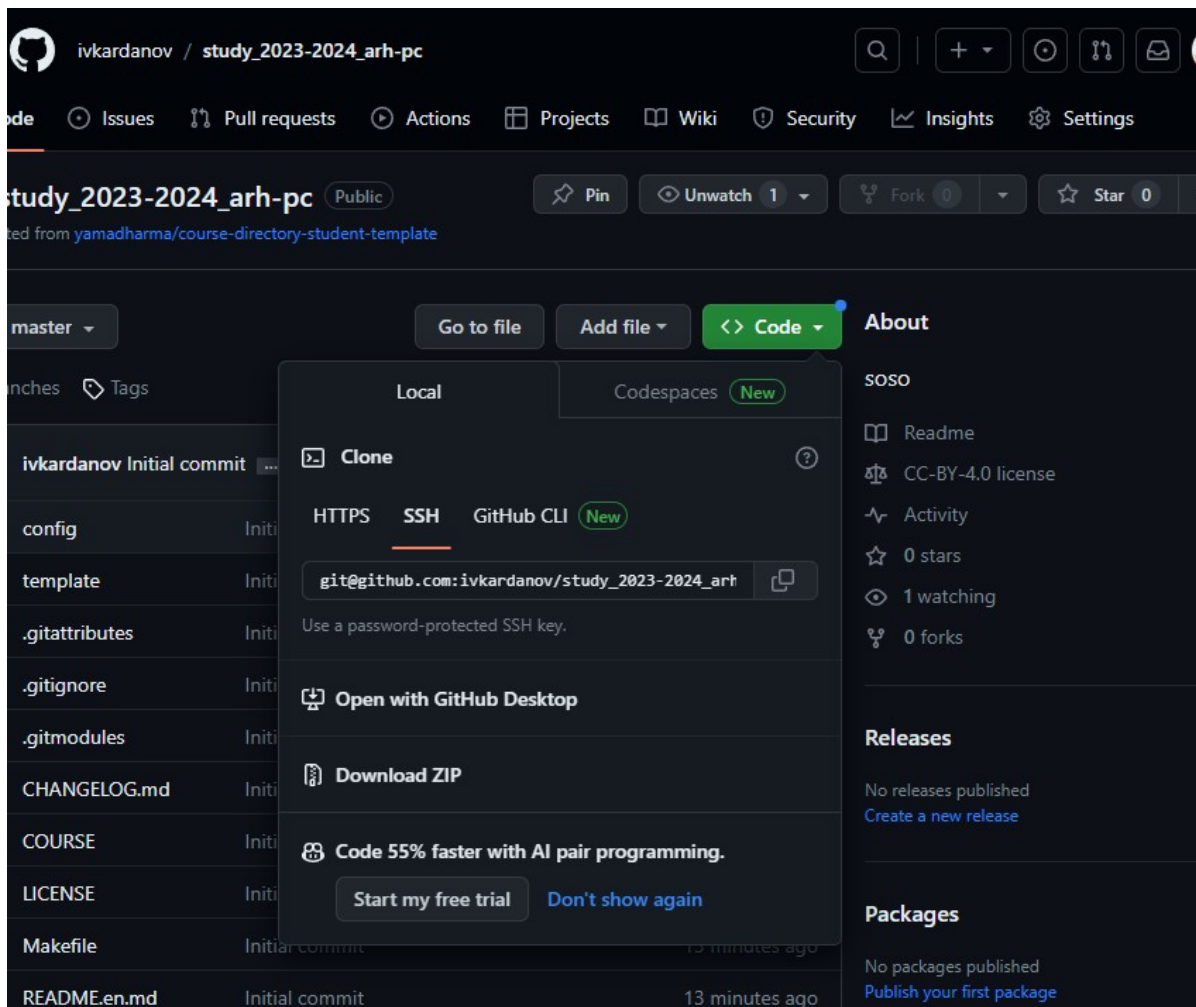
Далее выбираю «Use this template», чтобы использовать этот шаблон для своего репозитория (рис. ??)



{#fig:10

width = 70%]

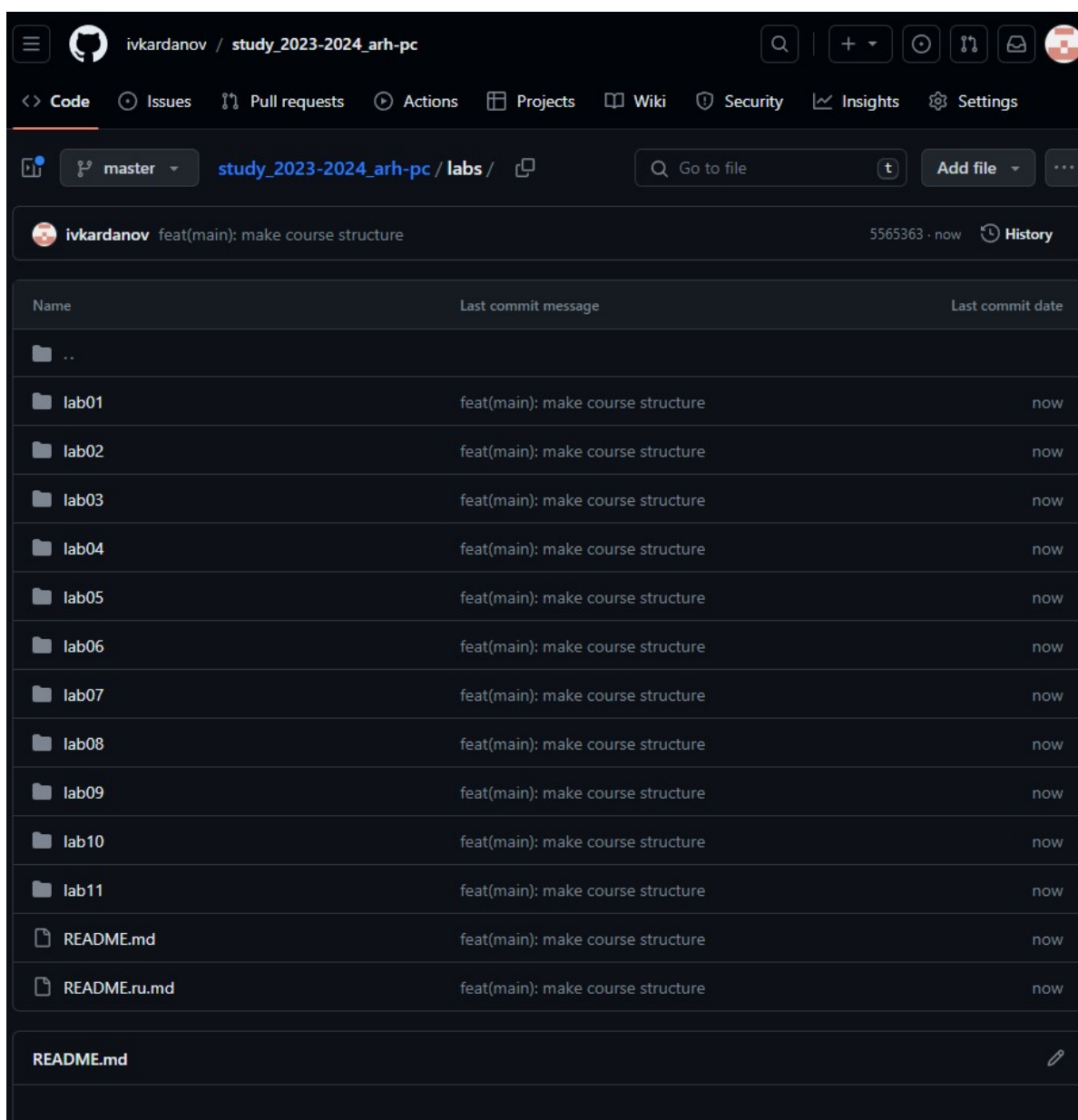
В открывшемся окне задаю имя репозитория: study\_2023–2024\_arh-pc. Создаю репозиторий, нажимаю на кнопку «Create repository from template» (рис. ??)



{#fig:11

width = 70%]

Ждем создания репозитория, затем он откроется (рис. ??)



{#fig:12

width = 70%]

Далее через терминал перехожу в созданный каталог курса с помощью утилиты `cd`. Клонировать созданный репозиторий с помощью команды: «`git clone –recursive git@github.com:/study_2023–2024_arh-pc.git arch-pc`» (рис. ??)

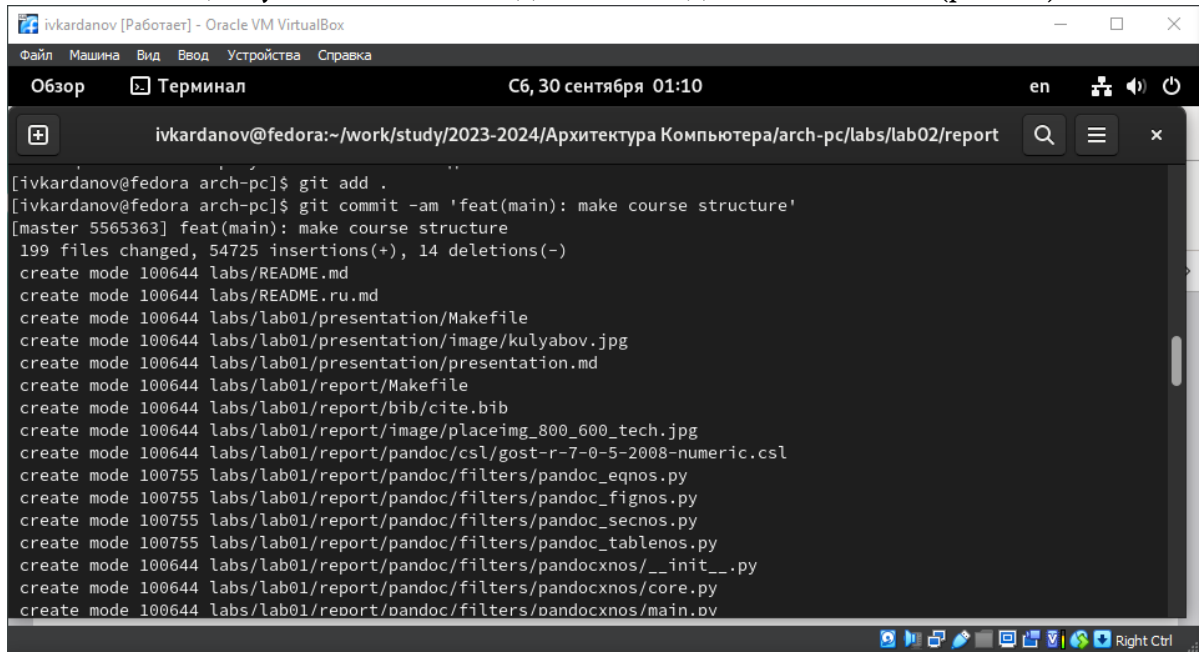
Клонирование репозитория{#fig:13 width = 70%]

Ссылку для копирования можно взять на странице созданного репозитория, сначала перейдя в окно «code», далее выбрав в окне вкладку «SSH» (рис. ??)

Окно с ссылкой на копирование репозитория{#fig:14 width = 70%]

## 6. Настройка каталога курса

Перехожу в каталог arch-pc с помощью утилиты cd. Затем удаляю лишние файлы с помощью утилиты rm и создаю необходимые каталоги (рис. ??)



```
ivkardanov [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Обзор  Терминал  С6, 30 сентября 01:10  en  [иконки]
ivkardanov@fedora:~/work/study/2023-2024/Архитектура Компьютера/arch-pc/labs/lab02/report

[ivkardanov@fedora arch-pc]$ git add .
[ivkardanov@fedora arch-pc]$ git commit -am 'feat(main): make course structure'
[master 5565363] 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
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
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/__init__.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/core.py
create mode 100644 labs/lab01/report/pandoc/filters/pandocxnos/main.py
```

width = 70%]

Отправляю созданные каталоги с локального репозитория на сервер: добавляю все созданные каталоги с помощью «git add», комментирую и сохраняю изменения на сервере с помощью «git commit» (рис. ??)

Добавление и сохранение изменений на сервере{#fig:16 width = 70%]

Отправляю все на сервер с помощью «push» (рис. ??)

Выгрузка изменений на сервер{#fig:17 width = 70%]

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

Проверка результата на странице репозитория{#fig:18 width = 70%]

## 7. Выполнение заданий для самостоятельной работы

Перехожу в директорию labs/lab02/report с помощью утилиты cd. Создаю в каталоге файл для отчета по третьей лабораторной работе с помощью утилиты touch. (рис. ??)

Перемещение по директориям, создание файла{#fig:19 width = 70%]

- 1) Оформить отчет я смогу в текстовом процессоре LibreOffice Writer, найдя его в меню приложений. После открытия текстового процессора открываю в нем созданный файл. Теперь можно создать в нем отчет (рис. ??)

Работа с отчётом в текстовом редакторе{#fig:20 width = 70%]

- 2) Перехожу из подкаталога lab02/report в подкаталог lab01/report с помощью утилиты cd. Проверяю местонахождение файла с отчетом по первой лабораторной работе, используя команду ls (рис. ??)

Перемещение между директориями, проверка местонахождения файла{#fig:21 width = 70%]

Перехожу из подкаталога lab01/report в подкаталог lab02/report с помощью утилиты cd. Копирую вторую лабораторную с помощью утилиты cp. (рис. ??)

Перемещение между директориями, копирование файла{#fig:20 width = 70%]

- 3) Добавляю с помощью команды git add в коммит созданные файлы:

Л01\_Кузьмин\_отчет. Перехожу в директорию, в которой находится отчет по второй лабораторной. Добавляю файл Л01\_Кузьмин\_отчет (рис. ??), (рис. ??)

```
[ivkardanov@fedora report]$ git push -f origin master
Перечисление объектов: 62, готово.
Подсчет объектов: 100% (62/62), готово.
Сжатие объектов: 100% (54/54), готово.
Запись объектов: 100% (62/62), 358.99 КиБ | 6.30 МиБ/с, готово.
Всего 62 (изменений 5), повторно использовано 26 (изменений 1), повторно использовано пакетов 0
remote: Resolving deltas: 100% (5/5), done.
To github.com:ivkardanov/study_2023-2024_arh-pc.git
+ 276b682...5565363 master -> master (forced update)
[ivkardanov@fedora report]$
```

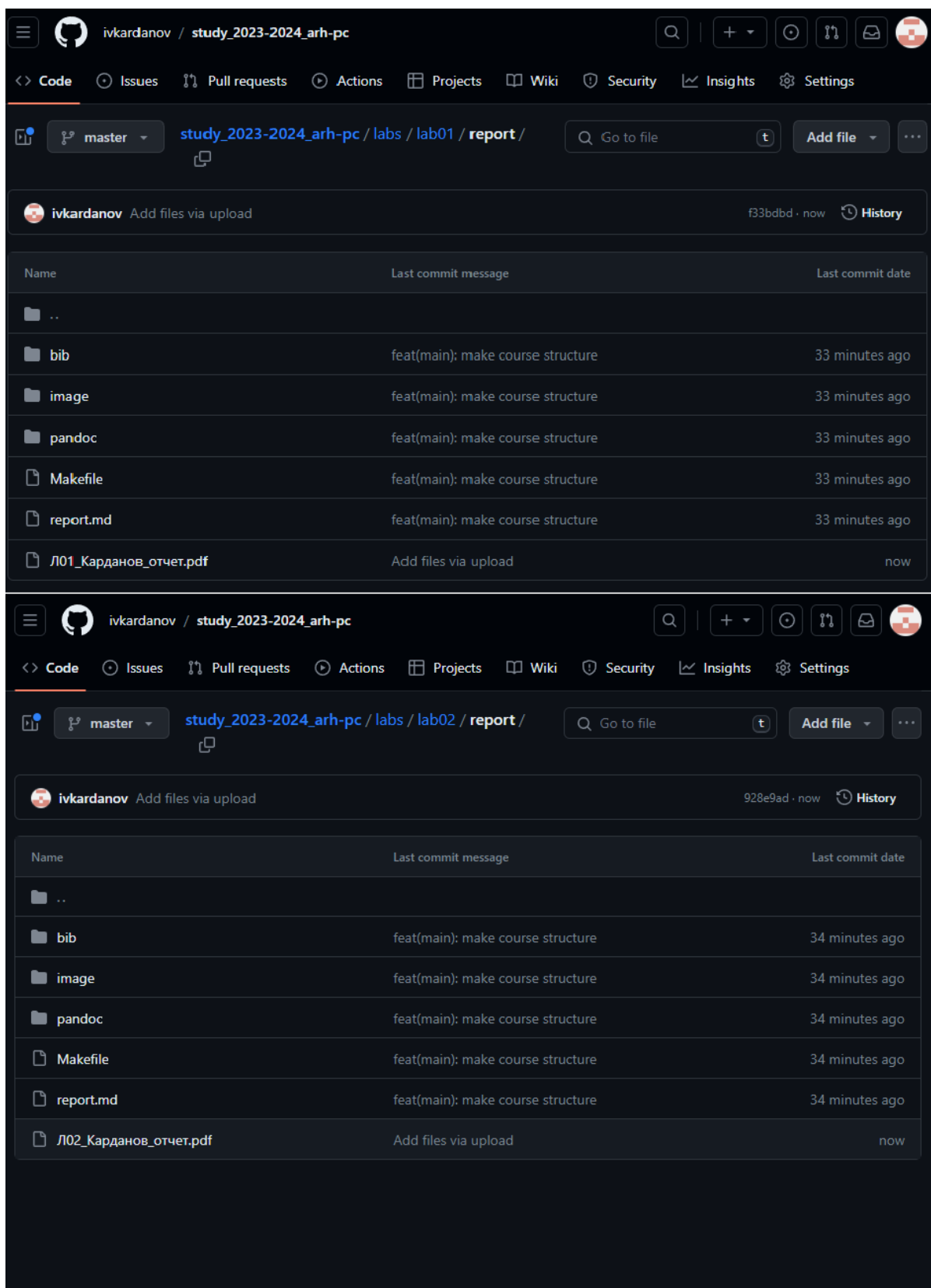
{#fig:23

width = 70%]

Добавление файла на сервер{#fig:20 width = 70%]

Отправляю в центральный репозиторий сохраненные изменения командой git push -f origin master (рис. ??)





{#fig:25



width = 70%]

Проверяю на сайте GitHub правильность выполнения заданий. Вижу, что пояснение к совершенным действиям отображается (рис. ??)

Отправка в центральный репозиторий сохраненных изменений{#fig:26 width = 70%]

При просмотре изменений так же вижу, что были добавлены файлы с отчетами по лабораторным работам (рис. ??)

Страница последних изменений в репозитории{#fig:27 width = 70%]

Вижу, что отчеты по лабораторным работам находятся в соответствующих каталогах репозитория: отчет по первой - в lab01/report, по второй – в lab02/report (рис. ??), (рис. ??)

Проверка каталогов{#fig:28 width = 70%]

Проверка каталогов{#fig:28 width = 70%]

## 5 Выводы

Я приобрел практический опыт работы с системой git, изучил принципы и применение контроля версий.

# Список литературы

1. Архитектура ЭВМ ([rudn.ru](http://rudn.ru))
2. Инструкция по использованию Git