

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №3.

СИСТЕМА КОНТРОЛЯ ВЕРСИЙ GIT

Дисциплина:      Архитектура компьютера

Студент: Сагдеров Камал Русланович

Группа: НКАбд-05-22

МОСКВА

2022 г.

# Содержание

1.Цель работы

2. Список иллюстраций

3.Теоретическое введение

4.Выполнение лабораторной работы

5.Выводы

### **3.1. Цель работы**

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

## **Список иллюстраций**

**[Рис1.Создание учетной записи](#)**

**[Рис2. Имя и email владельца репозитория](#)**

**[Рис3. Настройка utf-8 в git](#)**

**[Рис 4. Имя начальной ветки](#)**

**[Рис 5. Параметры autocrlf и safecrlf](#)**

**[Рис6. Сгенерирование приватного и открытого ключей](#)**

**[Рис7. Загрузка сгенерированного ключа](#)**

**[Рис8. Создание каталога](#)**

**[Рис9.Создание репозитория](#)**

**[Рис10. Переход в каталог курса](#)**

**[Рис11.Клонирование репозитория](#)**

**[Рис12. Настройка каталога курса](#)**

## **3.2. Теоретическое введение**

### **3.2.1. 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 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.

### **3.2.2. 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.

### **3.2.4. 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

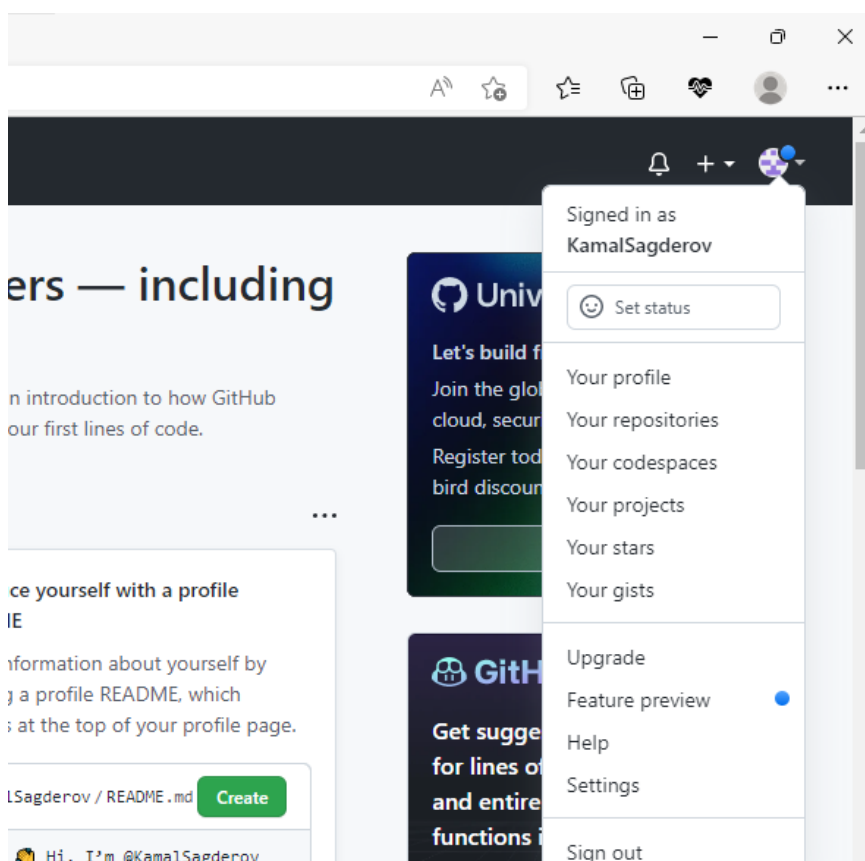
или

git push

### 3. Порядок выполнения лабораторной работы

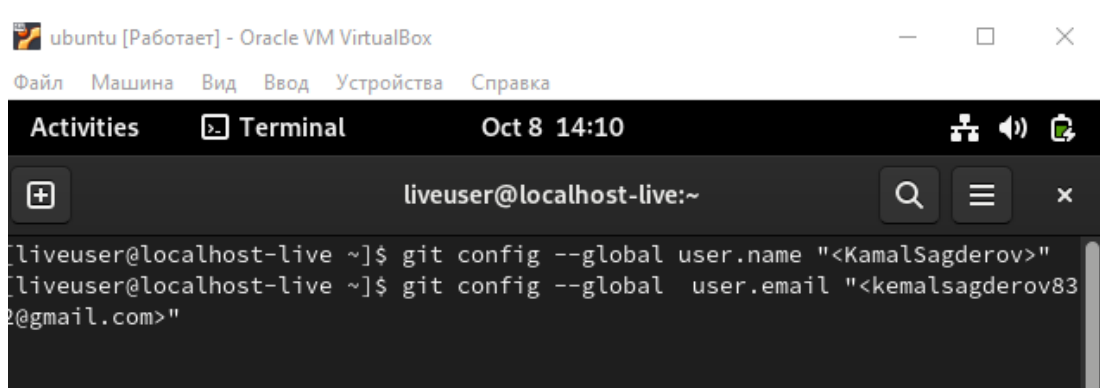
#### 3.1. Настройка github

Создаю учётную запись на сайте <https://github.com/> и заполняю основные данные.



**Рис1.**Создание учетной записи

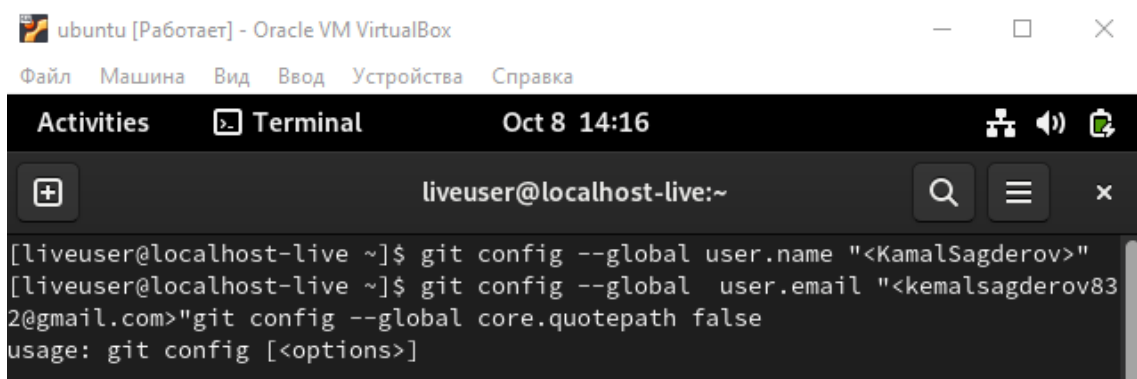
Открываю терминал и ввожу следующие команды, указав имя и email своего репозитория:



**Рис2.** Имя и email владельца репозитория



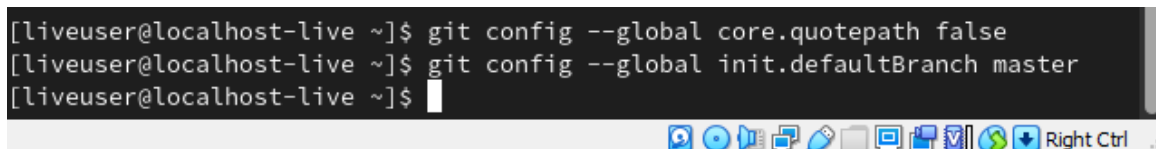
Настраиваю utf-8 в выводе сообщений git:



```
ubuntu [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка
Activities  Terminal  Oct 8 14:16
liveuser@localhost-live:~
[liveuser@localhost-live ~]$ git config --global user.name "<KamalSagderov>"
[liveuser@localhost-live ~]$ git config --global user.email "<kemalsagderov832@gmail.com>"
[liveuser@localhost-live ~]$ git config --global core.quotePath false
usage: git config [<options>]
```

**Рис3.** Настройка utf-8 в git

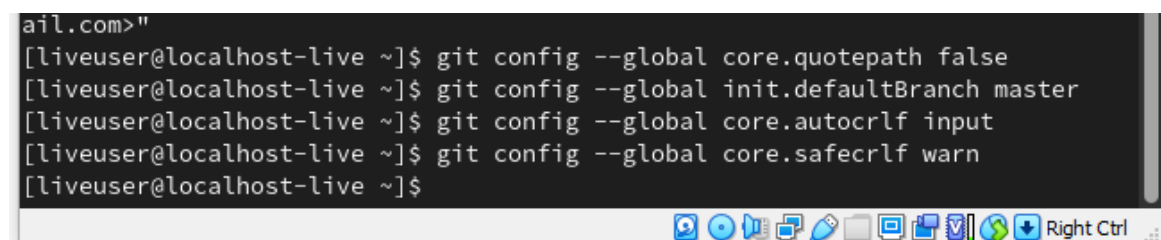
Задаю имя начальной ветки (будем называть её master):



```
[liveuser@localhost-live ~]$ git config --global core.quotePath false
[liveuser@localhost-live ~]$ git config --global init.defaultBranch master
[liveuser@localhost-live ~]$
```

**Рис 4.** Имя начальной ветки

Параметры autocrlf и safecrlf:

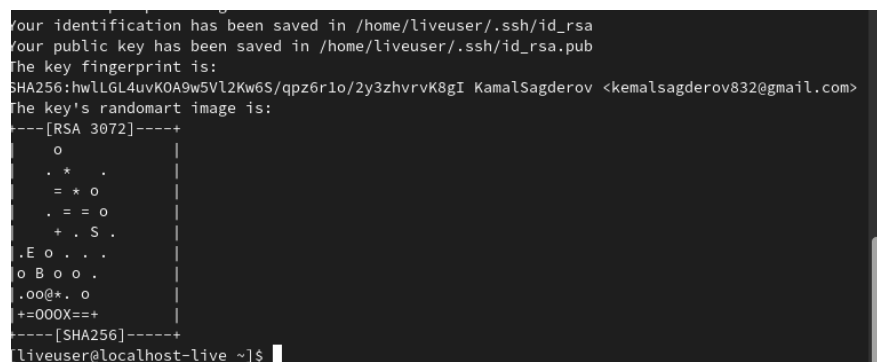


```
ail.com>"
[liveuser@localhost-live ~]$ git config --global core.quotePath false
[liveuser@localhost-live ~]$ git config --global init.defaultBranch master
[liveuser@localhost-live ~]$ git config --global core.autocrlf input
[liveuser@localhost-live ~]$ git config --global core.safecrlf warn
[liveuser@localhost-live ~]$
```

**Рис 5.** Параметры autocrlf и safecrlf

### 3.4.3. Создание SSH ключа

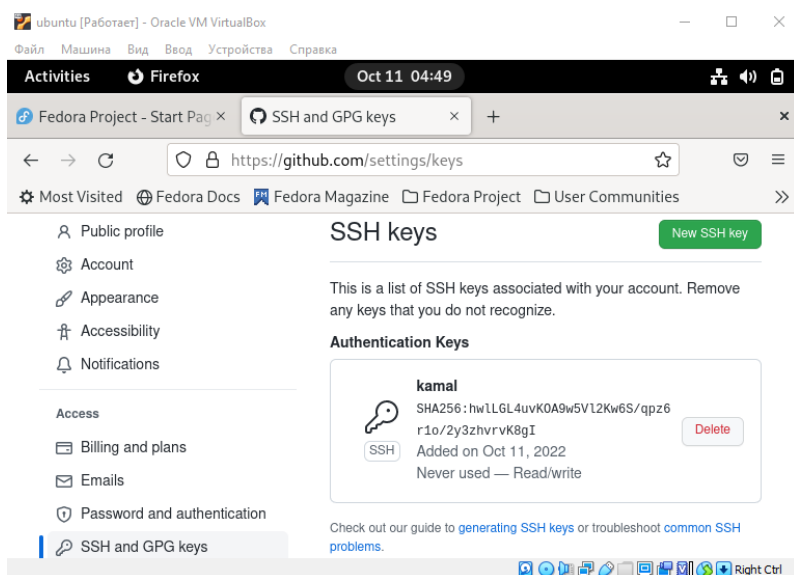
Для своей идентификации на сервере репозитория сгенерирую пару ключей (приватный и открытый):



```
Your identification has been saved in /home/liveuser/.ssh/id_rsa
Your public key has been saved in /home/liveuser/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:hwLLGL4uvKOA9w5Vl2Kw6S/qpz6r1o/2y3zhvrvK8gI KamalSagderov <kemalsagderov832@gmail.com>
The key's randomart image is:
----[RSA 3072]-----+
  o
  . * .
  = * o
  . = = o
  + . S .
  .E o . . .
  o B o o .
  .oo@* . o
  +=000X==+
-----[SHA256]-----+
[liveuser@localhost-live ~]$
```

**Рис6.** Сгенерирование приватного и открытого ключей

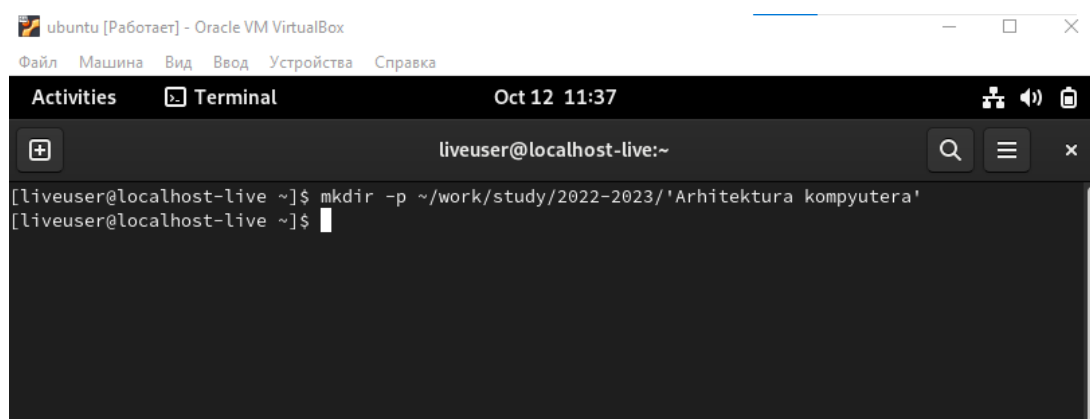
Ключи сохраняться в каталоге ~/.ssh/. Далее необходимо загрузить сгенерённый открытый ключ. Для этого зайти на сайт <http://github.org/> под своей учётной записью и перейти в меню Setting . После этого выбираю в боковом меню SSH and GPG keys и нажал кнопку New SSH key . Скопировав из локальной консоли ключ в буфер обмена



**Рис7.** Загрузка сгенерированного ключа

#### 3.4.4. Сознание рабочего пространства и репозитория курса на основе шаблона

Открываю терминал и создаю каталог для предмета «Архитектура компьютера»



**Рис8.** Создание каталога

### 3.4.5. Сознание репозитория курса на основе шаблона

Репозиторий на основе шаблона можно создать через web-интерфейс github.

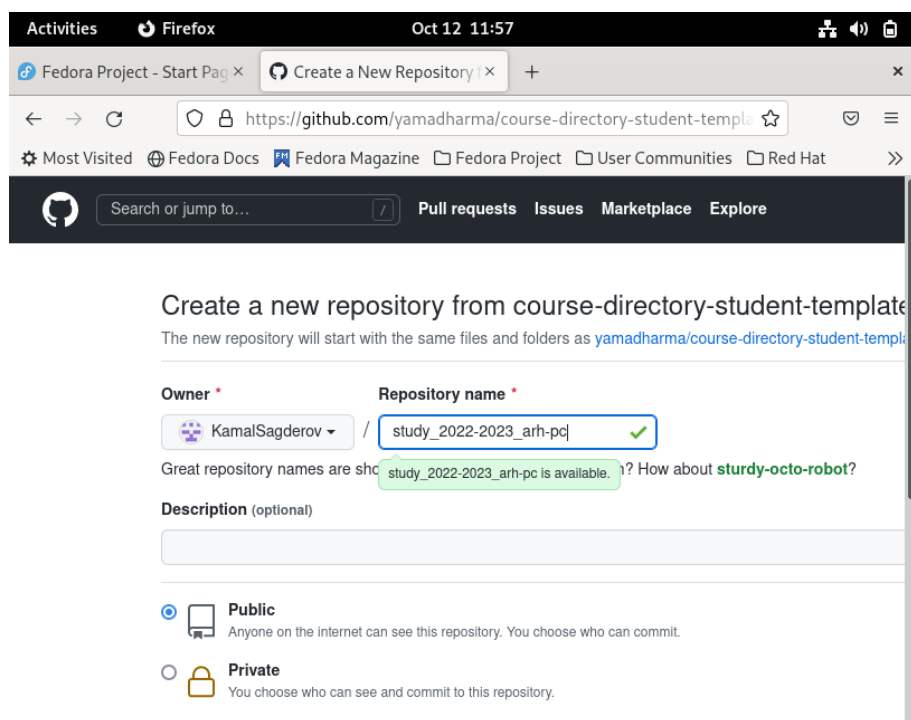


Рис9.Создание репозитория

Открываю терминал и перехожу в каталог курса:

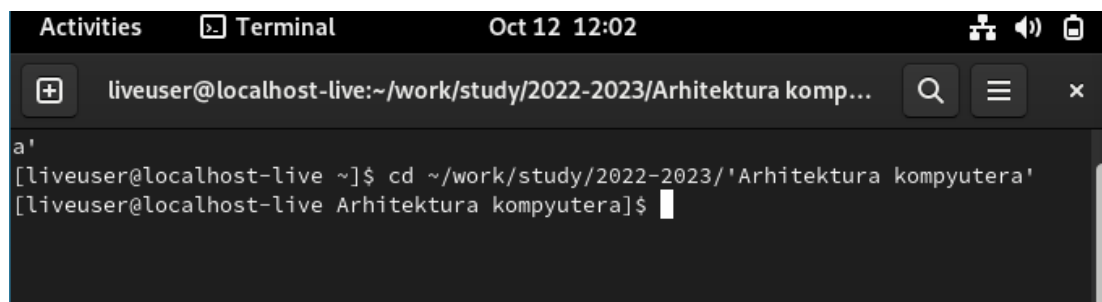
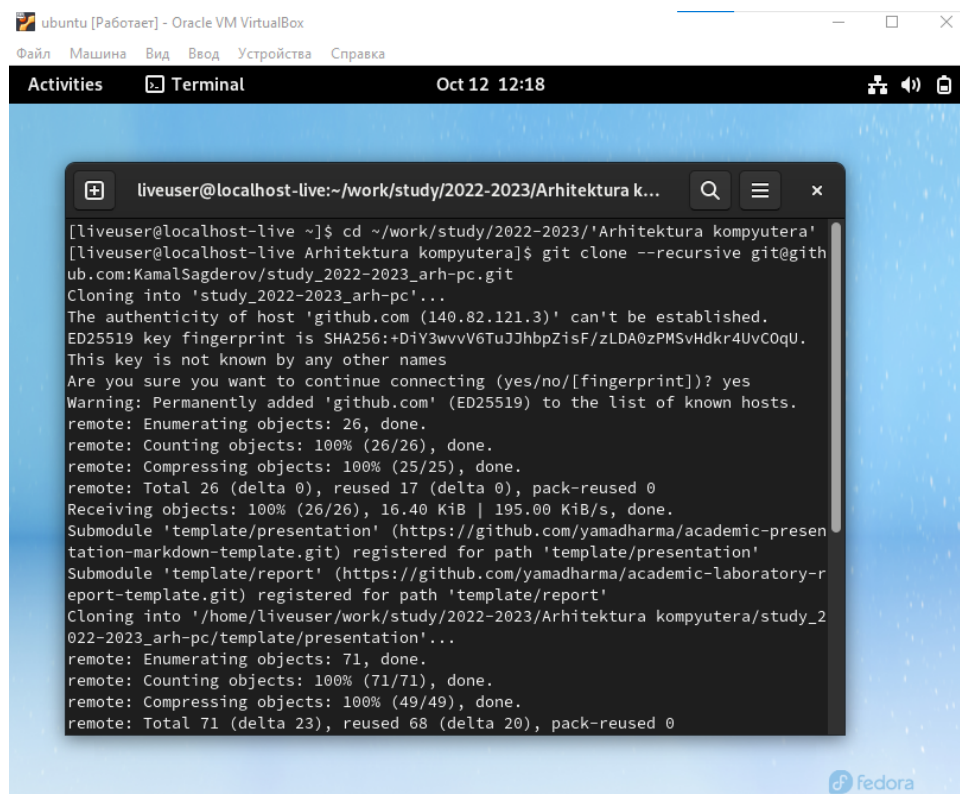


Рис10. Переход в каталог курса

клонирую созданный репозиторий:



The screenshot shows a terminal window titled "liveuser@localhost-live:~/work/study/2022-2023/Arhitektura k...". The terminal output shows the user navigating to a directory and running the command `git clone --recursive git@github.com:KamalSagderov/study_2022-2023_arh-pc.git`. The output includes a warning about the host's authenticity, a confirmation to continue, and the progress of cloning the repository and its submodules. The submodules are `template/presentation` and `template/report`. The cloning process is completed successfully.

```
[liveuser@localhost-live ~]$ cd ~/work/study/2022-2023/Arhitektura kompyutera
[liveuser@localhost-live Arhitektura kompyutera]$ git clone --recursive git@github.com:KamalSagderov/study_2022-2023_arh-pc.git
Cloning into 'study_2022-2023_arh-pc'...
The authenticity of host 'github.com (140.82.121.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvC0qU.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
remote: Enumerating objects: 26, done.
remote: Counting objects: 100% (26/26), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 26 (delta 0), reused 17 (delta 0), pack-reused 0
Receiving objects: 100% (26/26), 16.40 KiB | 195.00 KiB/s, done.
Submodule 'template/presentation' (https://github.com/yamadharm/academic-presentation-markdown-template.git) registered for path 'template/presentation'
Submodule 'template/report' (https://github.com/yamadharm/academic-laboratory-report-template.git) registered for path 'template/report'
Cloning into '/home/liveuser/work/study/2022-2023/Arhitektura kompyutera/study_2022-2023_arh-pc/template/presentation'...
remote: Enumerating objects: 71, done.
remote: Counting objects: 100% (71/71), done.
remote: Compressing objects: 100% (49/49), done.
remote: Total 71 (delta 23), reused 68 (delta 20), pack-reused 0
```

Рис11.Клонирование репозитория

### 3.4.6. Настройка каталога курсаR

Комментарий: Перехожу в каталог курса и удаляю лишние файлы: `rm package.json`, создаю необходимые каталоги: `echo arch-pc > COURSE` `make`, отправляю файлы на сервер: `git add .` `git commit -am 'feat(main): make course structure'` `git push`

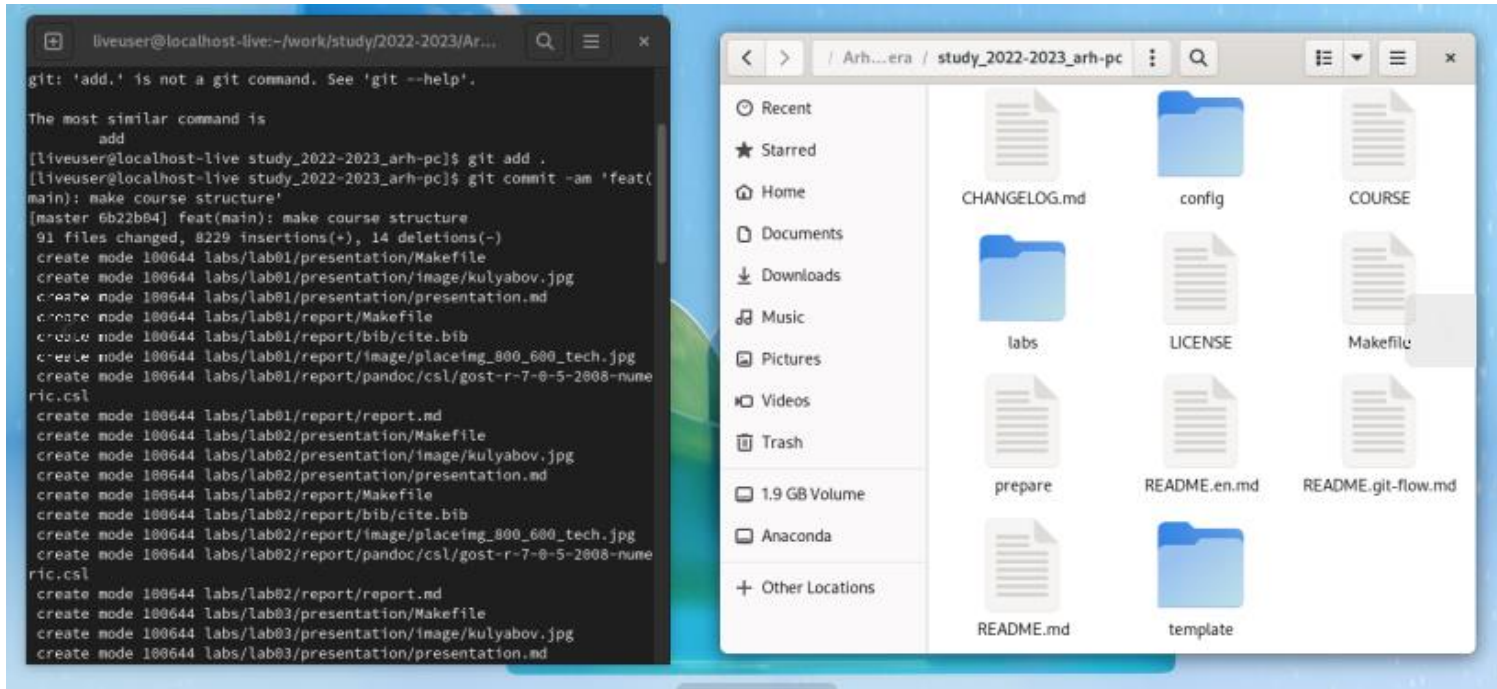
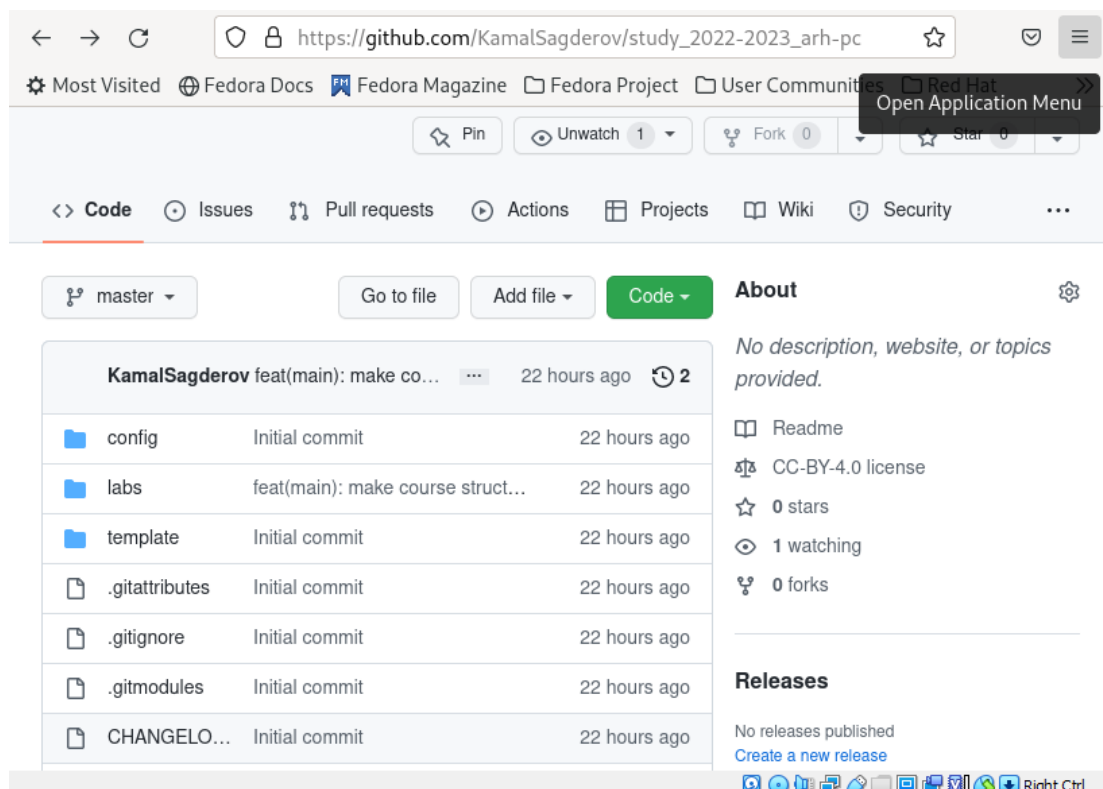


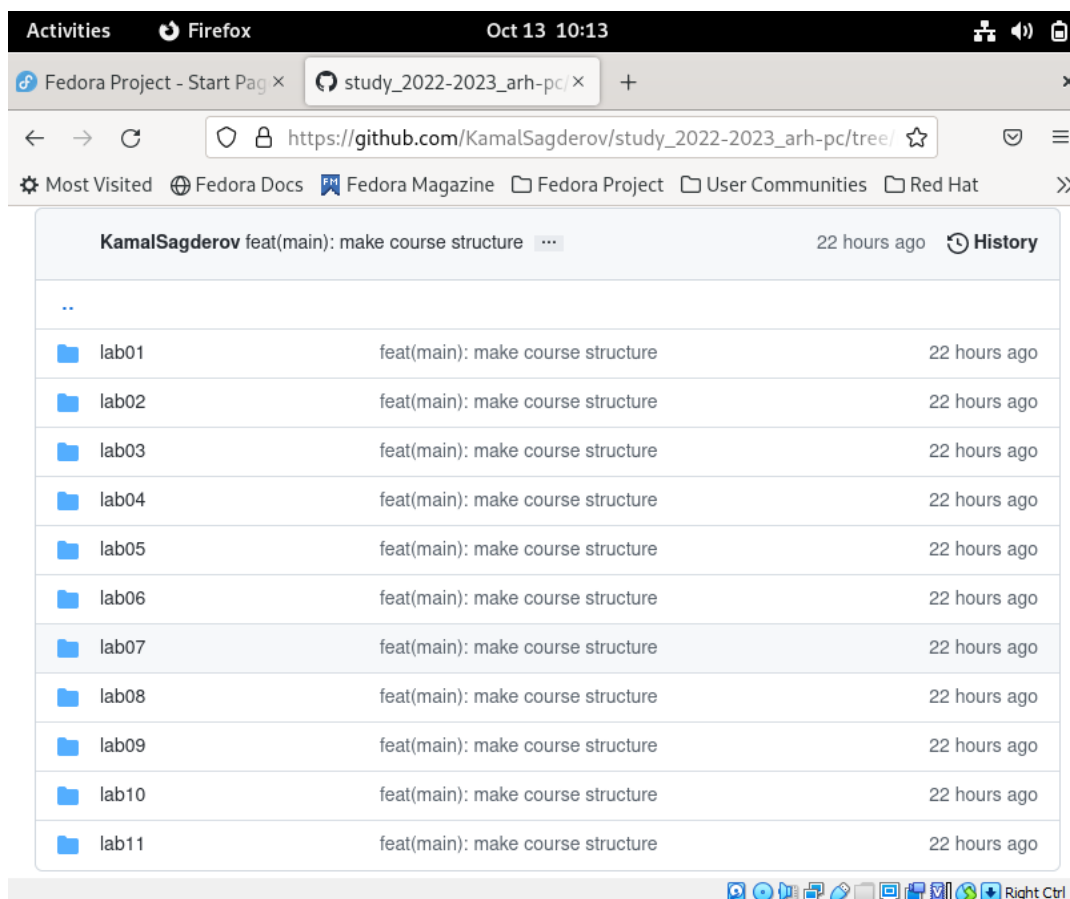
Рис12. Настройка каталога курса



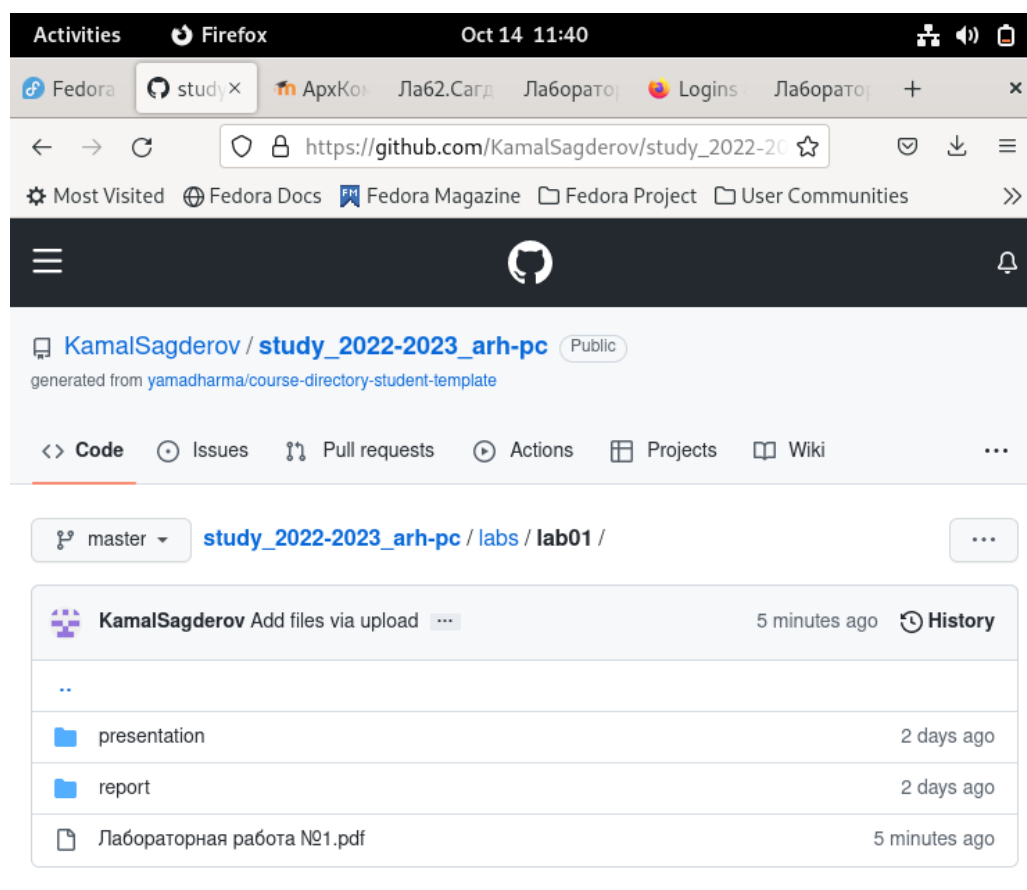
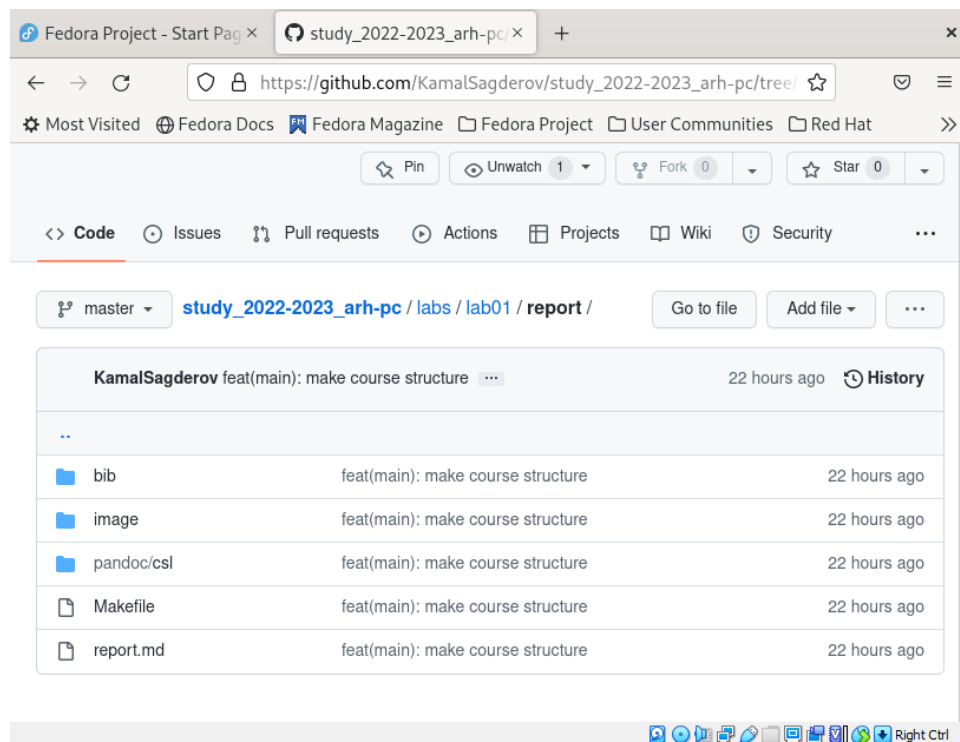
**Комментарий1:** Захожу на свой репозиторий в github, вижу что появились все необходимые файлы и все работает.

### 3.5. Задание для самостоятельной работы

1. Создайте отчет по выполнению лабораторной работы в соответствующем каталоге рабочего пространства (labs>lab03>report)



**Комментарий 2:** На github в моем репозитории появились все необходимые, 11 папок lab



Комментарий 3 : Загрузил первую лабораторную работу №1 и точно также лабораторную работу №2 на github.

**Вывод:** После завершения лабораторной работы №3, я приобрел практические способности по работе с системой git