

# Лабораторная работа №1

## Дисциплина: Основы информационной безопасности

Колчева Юлия Вячеславовна

### Содержание

Цель работы .....	1
Задание .....	1
Выполнение лабораторной работы .....	1
Выводы.....	1515

### Цель работы

Цель данной лабораторной работы — Изучить идеологию и применение средств контроля версий. Настроить Git. Научиться оформлять отчёты с помощью легковесного языка разметки Markdown .

### Задание

Часть 1 (Установка VitrualBox) 1.Установка и настройка Виртуальной машины VirtualBox (OC Linux)

Часть 2 (Работа с GitHub) 1.Создать базовую конфигурацию для работы с git. 2.Создать ключ SSH. 3.Создать ключ PGP. 4.Настроить подписи git. 5.Зарегистрироваться на GitHub. 6.Создать локальный каталог для выполнения заданий по предмету.

Часть 3 (Создание Markdown файла) 1.Сделайте отчёт по предыдущей лабораторной работе в формате Markdown. 2.В качестве отчёта предоставить отчёты в 3 форматах: pdf,docx и md

### Выполнение лабораторной работы

Часть 1. Установка и настройка Виртуальной машины VirtualBox

1). Для создания виртуальной машины используем программу Oracle VM VirtualBox. Для начала нужно настроить месторасположение виртуальной машины

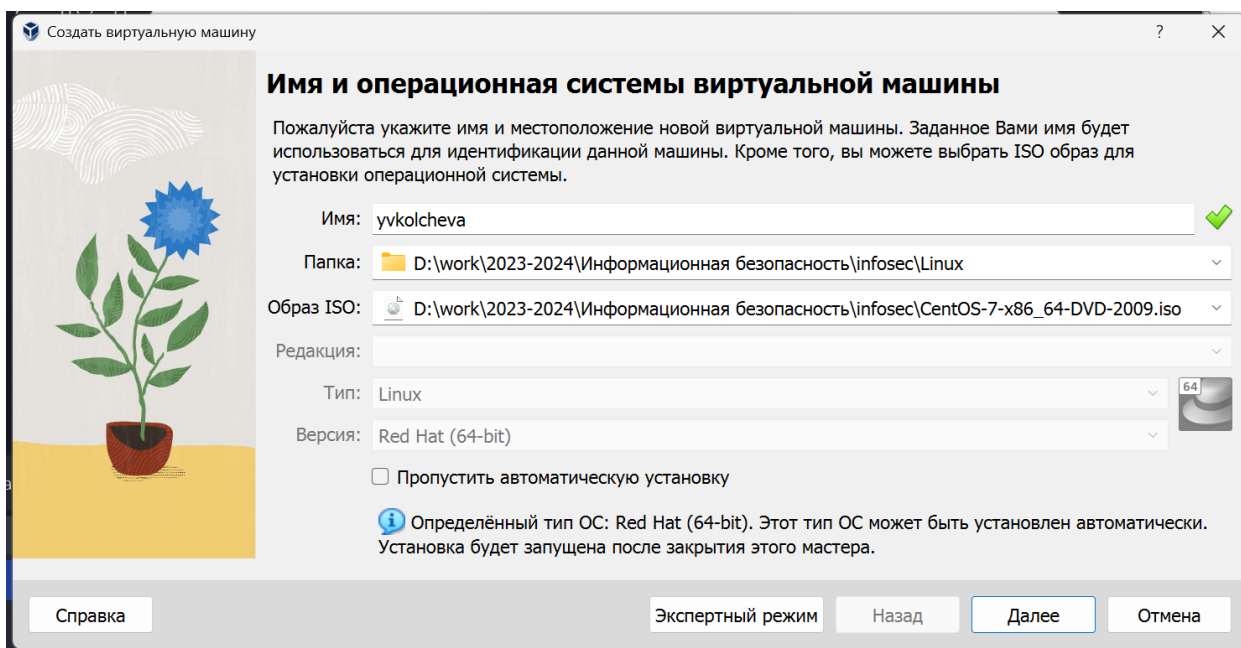


Figure 1: Установка

2). Создаём виртуальную машину с помощью кнопки “Создать”, вводим имя виртуальной машины и выбираем версию ОС, в данном случае Red Hat (64-bit)

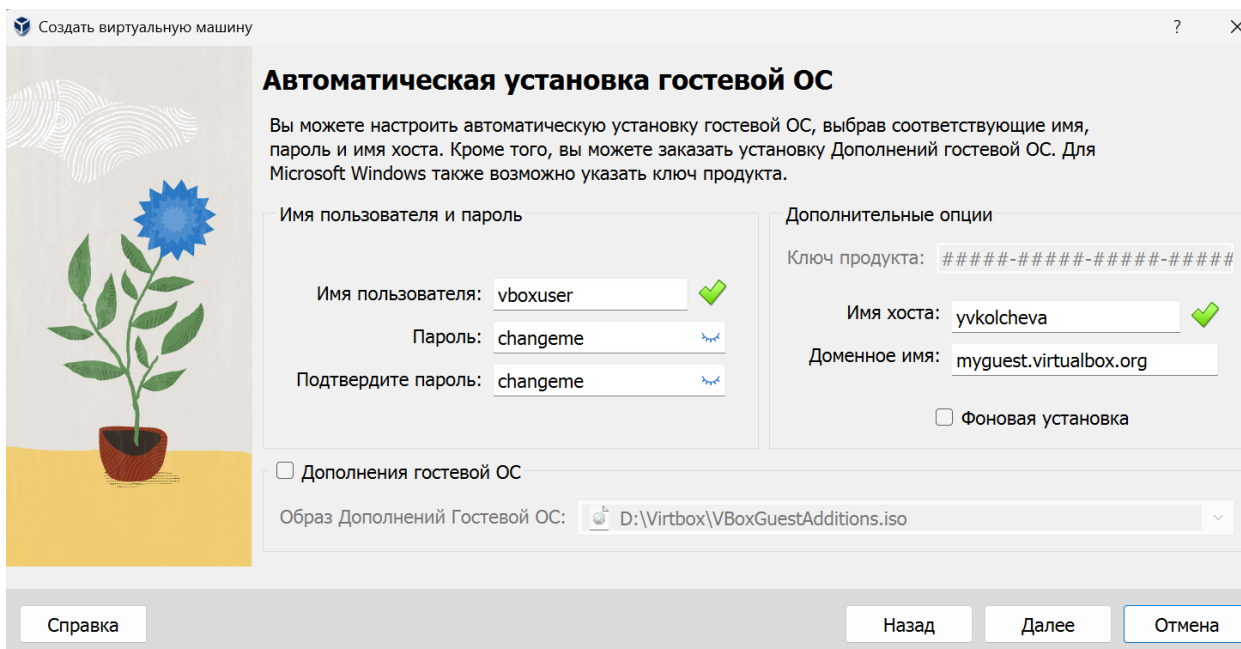


Figure 2: Автоматическая установка

3). Затем устанавливаем объем оперативной памяти — 2048 МБ

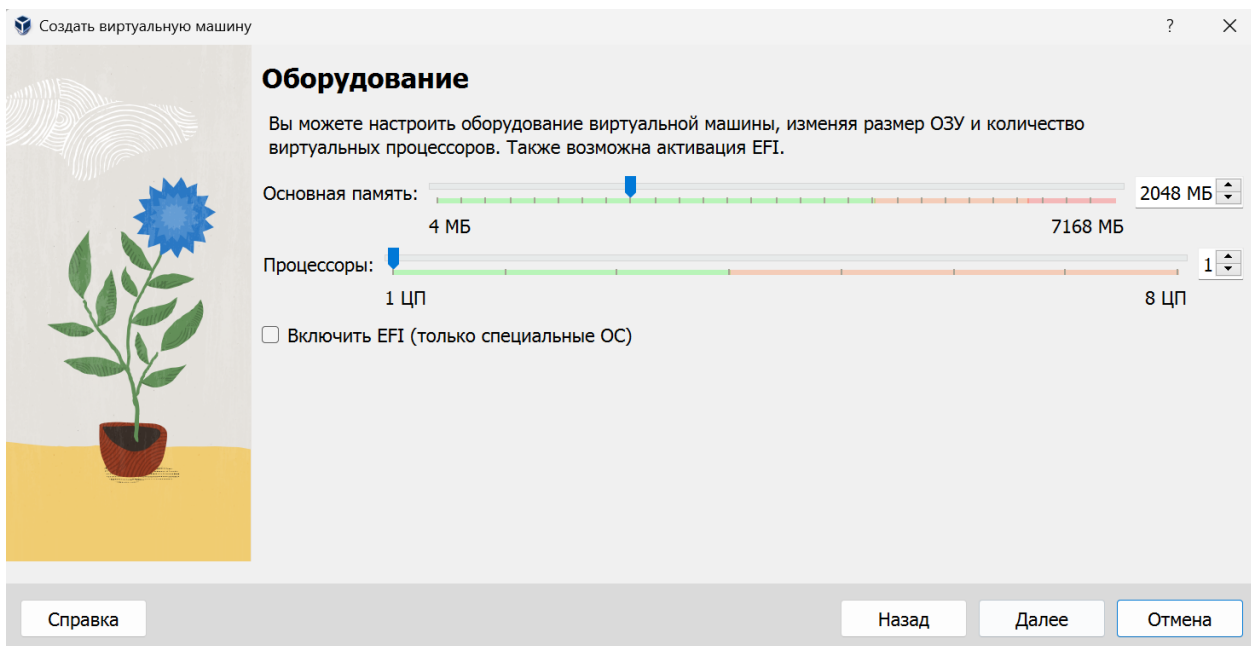


Figure 3: Установка объёма оперативной памяти

4). Создаём новый виртуальный жёсткий диск. А также назначаем размер жёсткого диска — 40 ГБ

5). Настраиваем виртуальную машину. В разделе “Носители” выбираем новый оптический диск, в данном случае это Rocky-8.6-x86\_64-dvd1.iso

6). Запускаем виртуальную машину

В разделе выбора программ добавляем в качестве дополнения Средства разработки

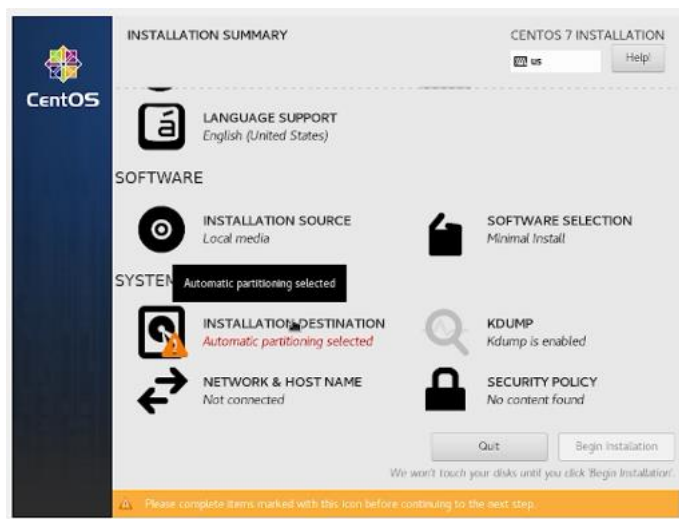


Figure 4: Настройка

Устанавливаем пароль для root и администратора. Создаём пользователя с правами администратора.

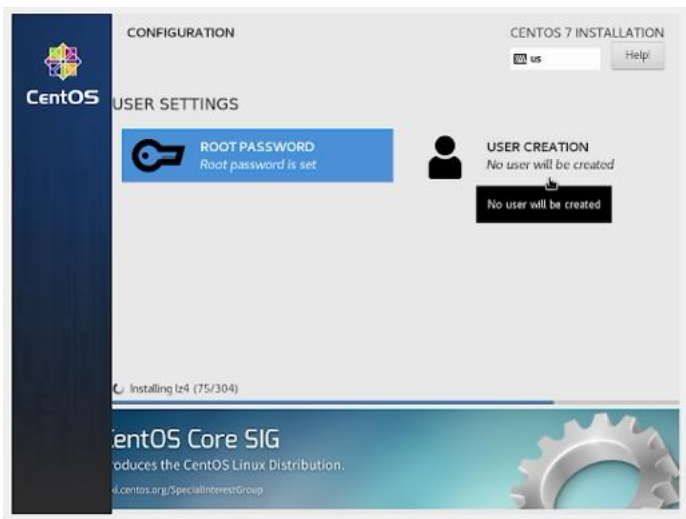


Figure 5: Настройка

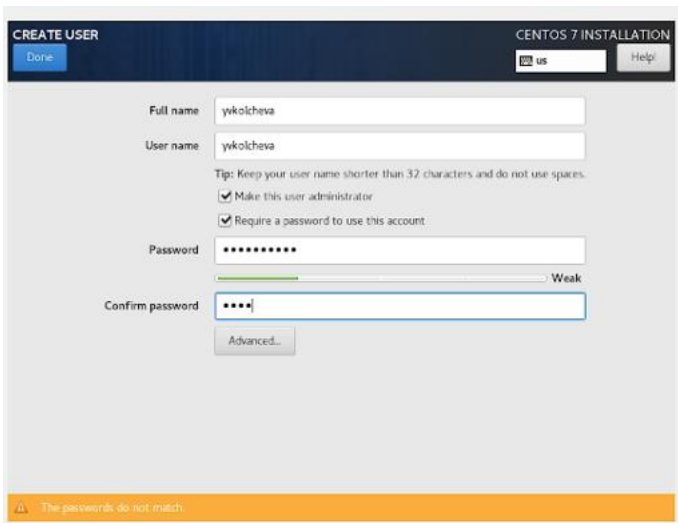
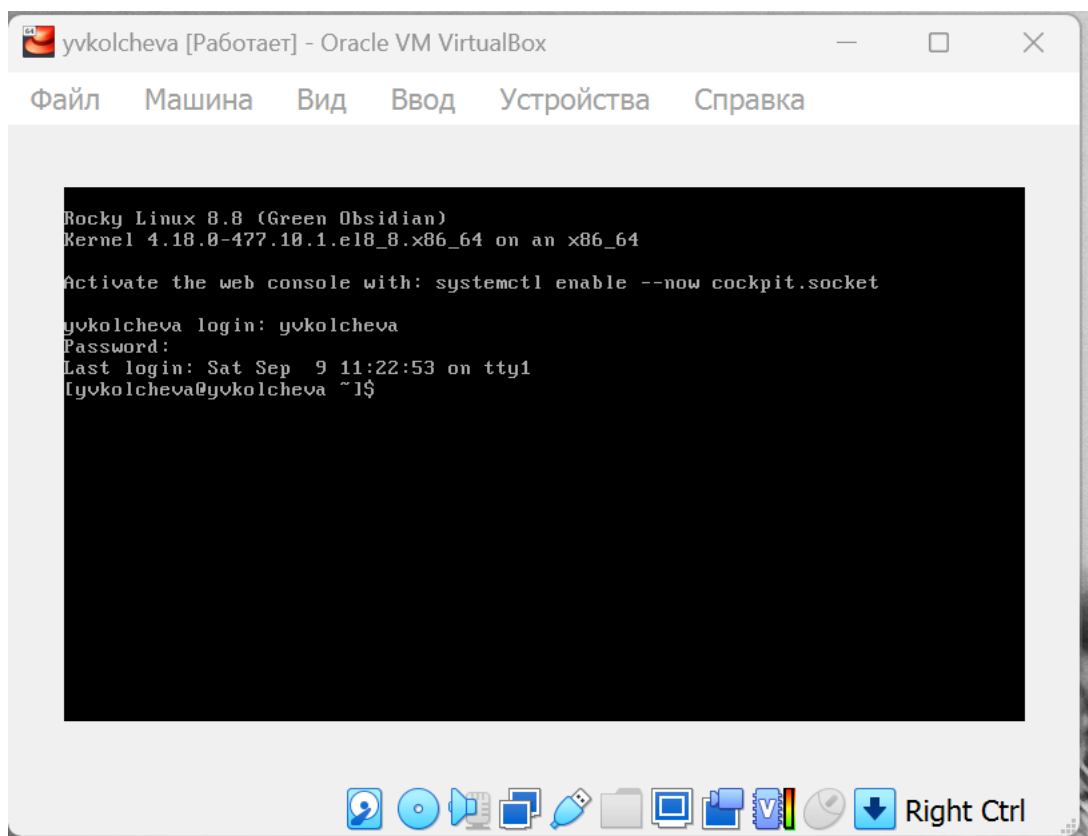


Figure 6: Настройка

Перезапускаем виртуальную машину. Дожидаемся загрузки системы и открываем консоль.



*Figure 7: Установка*

#### ДОМАШНЕЕ ЗАДАНИЕ №1:

- 1). В окне терминала проанализировала последовательность загрузки системы, выполнив команду `dmesg`. Можно просто просмотреть вывод этой команды: `dmesg | less`

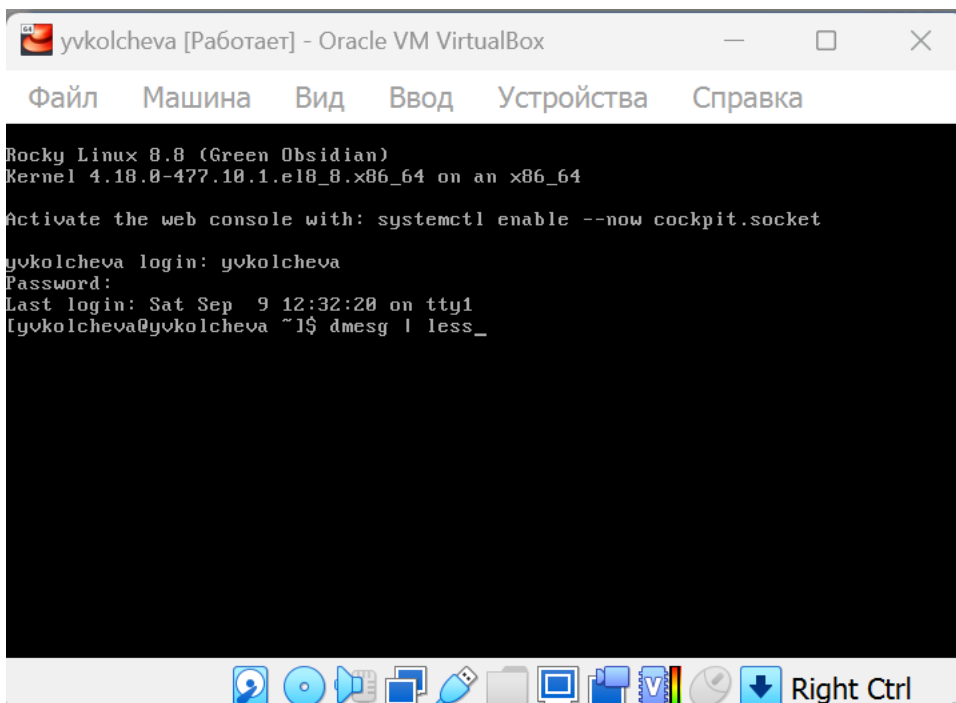


Figure 8: Команда dmesg

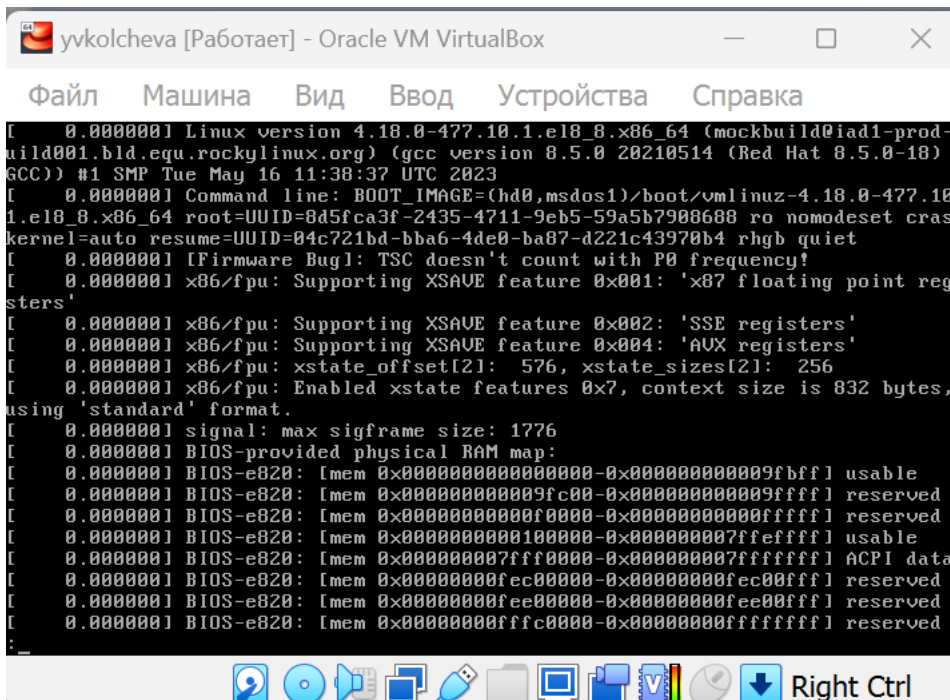


Figure 9: Команда dmesg

Можно использовать поиск с помощью `grep:dmesg | grep -i "то, что ищем"` а. Версия ядра Linux (Linux version).

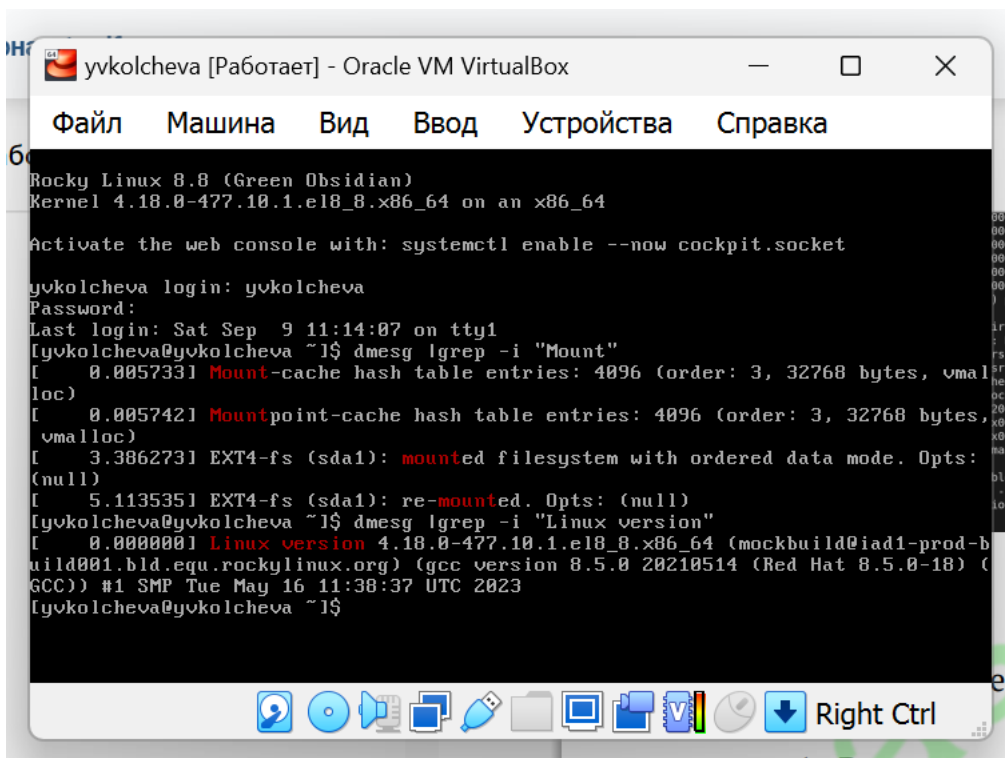


Figure 10: Версия ядра Linux

- b. Частота процессора (Detected Mhz processor).
- c. Модель процессора (CPU0)
- d. Тип обнаруженного гипервизора (Hypervisor detected)

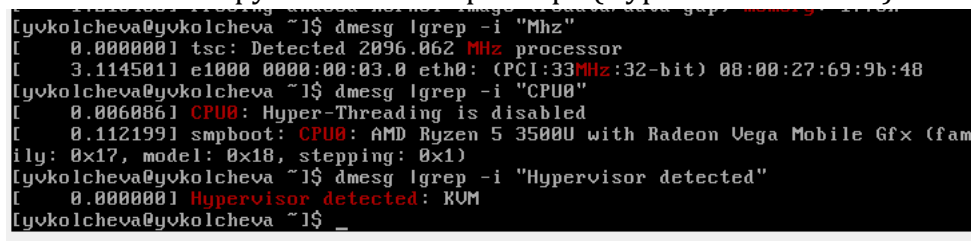
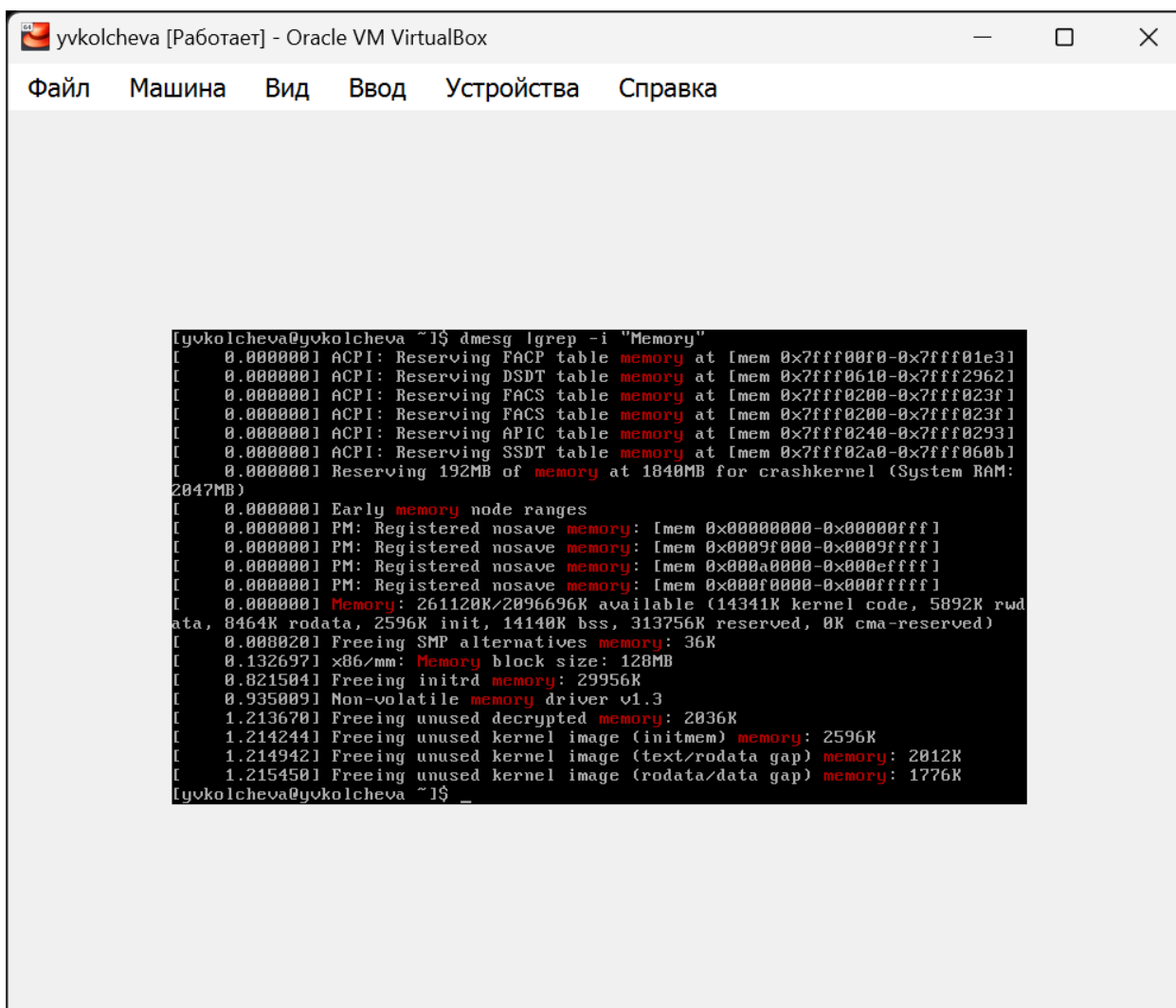


Figure 11: Частота процессора, модель процессора, объем доступной оперативной памяти

- d. Объем доступной оперативной памяти (Memory available)



The screenshot shows a terminal window titled "yvkolcheva [Работает] - Oracle VM VirtualBox". The menu bar includes "Файл", "Машина", "Вид", "Ввод", "Устройства", and "Справка". The terminal output displays the command `dmesg | grep -i "Memory"` and its results, which include ACPI table reservations, early memory node ranges, PM registered nosave memory, and the final available memory calculation.

```
[yvkolcheva@yvkolcheva ~]$ dmesg | grep -i "Memory"
[ 0.000000] ACPI: Reserving FACP table memory at [mem 0x7fff00f0-0x7fff01e3]
[ 0.000000] ACPI: Reserving DSDT table memory at [mem 0x7fff0610-0x7fff2962]
[ 0.000000] ACPI: Reserving FACS table memory at [mem 0x7fff0200-0x7fff023f]
[ 0.000000] ACPI: Reserving FACS table memory at [mem 0x7fff0200-0x7fff023f]
[ 0.000000] ACPI: Reserving APIC table memory at [mem 0x7fff0240-0x7fff0293]
[ 0.000000] ACPI: Reserving SSDT table memory at [mem 0x7fff02a0-0x7fff060b]
[ 0.000000] Reserving 192MB of memory at 1840MB for crashkernel (System RAM: 2047MB)
[ 0.000000] Early memory node ranges
[ 0.000000] PM: Registered nosave memory: [mem 0x00000000-0x00000fff]
[ 0.000000] PM: Registered nosave memory: [mem 0x0009f000-0x0009ffff]
[ 0.000000] PM: Registered nosave memory: [mem 0x000a0000-0x000effff]
[ 0.000000] PM: Registered nosave memory: [mem 0x000f0000-0x000fffff]
[ 0.000000] Memory: 261120K/2096696K available (14341K kernel code, 5892K rwd
ata, 8464K rodata, 2596K init, 14140K bss, 313756K reserved, 0K cma-reserved)
[ 0.000020] Freeing SMP alternatives memory: 36K
[ 0.132697] x86/mm: Memory block size: 128MB
[ 0.821504] Freeing initrd memory: 29956K
[ 0.935009] Non-volatile memory driver v1.3
[ 1.213670] Freeing unused decrypted memory: 2036K
[ 1.214244] Freeing unused kernel image (initmem) memory: 2596K
[ 1.214942] Freeing unused kernel image (text/rodata gap) memory: 2012K
[ 1.215450] Freeing unused kernel image (rodata/data gap) memory: 1776K
[yvkolcheva@yvkolcheva ~]$ _
```

Figure 12: Объем доступной оперативной памяти

- f. Тип файловой системы корневого раздела. Последовательность монтирования файловых систем



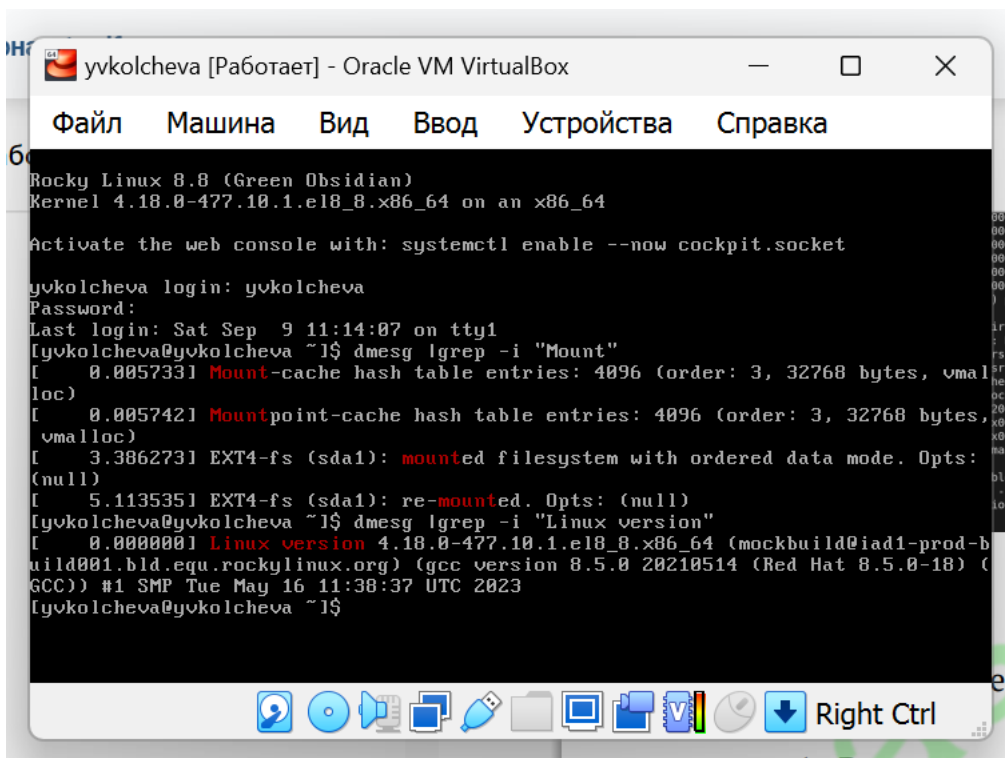


Figure 12: Mount

## КОНТРОЛЬНЫЕ ВОПРОСЫ №1

1).Учётная запись пользователя содержит сведения, необходимые для идентификации пользователя при подключении к системе, такие как имя пользователя, имя хоста и пароль. 2).Команды терминала: а.Для получения справки используется ключ `-help` или команда `man`. Например, `ls -help` или `man ls`. б.Для перемещения по файловой системе используется команда `cd`. Например `cd ~`. с.Для просмотра содержимого каталога используется команда `ls`. Например `ls ~/work`. д.Для определения объёма каталога используется команда `du`. е.Для создания каталогов используется `mkdir`, для удаления пустых каталогов используется `rmdir`. Для создания файлов используется `touch`, для удаления файлов и каталог используется `rm`. ф.Для задания прав используется команда `chmod`. Например, `chmod u-w test.txt`. г.Для просмотра истории команд используется команда `history`. 3).Файловая система — часть ОС, которая обеспечивает чтение и запись файлов на дисковых носителях информации. а.Ext2 — расширенная файловая система. Данные сначала кэшируются и только потом записываются на диск. б.Ext3 и Ext4 — журналируемые файловые системы. Осуществляется хранение в виде журнала со списком изменений, что помогает сохранить целостность при сбоях. с.XFS — высокопроизводительная журналируемая файловая система, рассчитанная для работы на дисках большого объёма. 4).Для просмотра подмонтированных в ОС файловых систем необходимо использовать команду `findmnt`. 5).Для удаления зависшего процесса используется команда `kill PID` или `killall название`.

## Часть 2. Работа с GitHub

1). Создаем учетную запись на <https://github.com>.

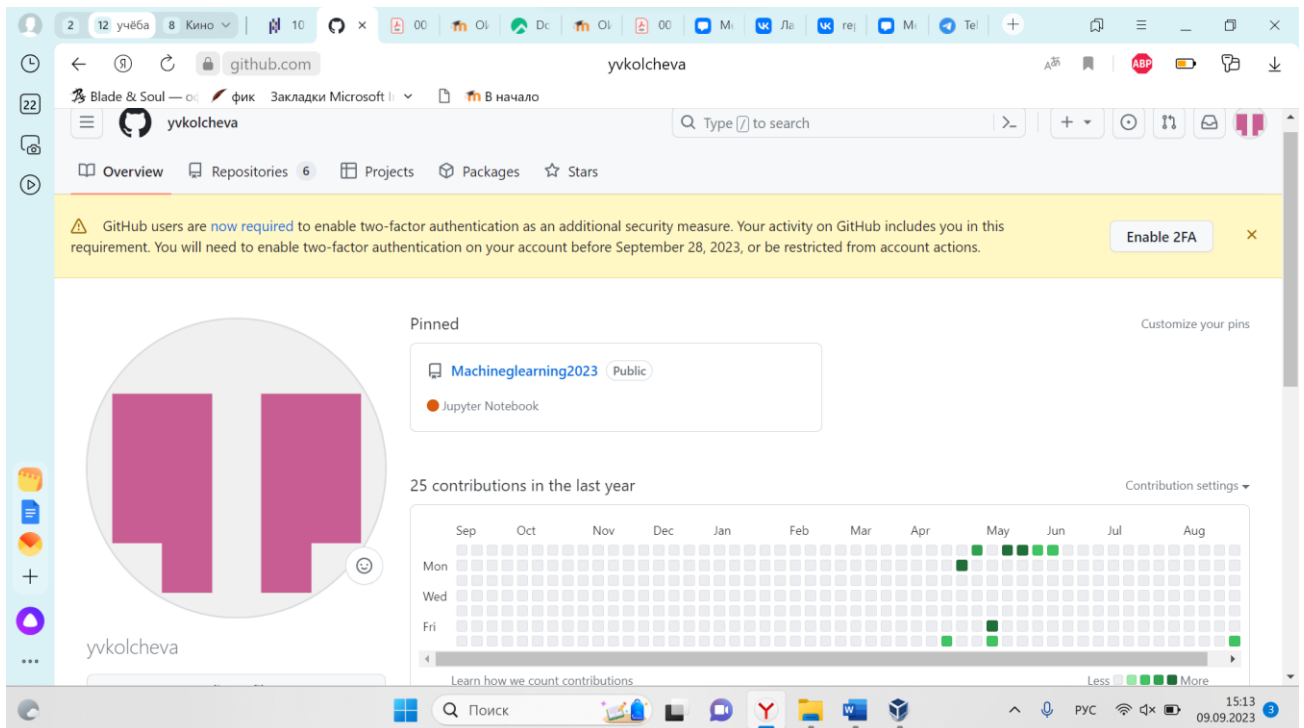
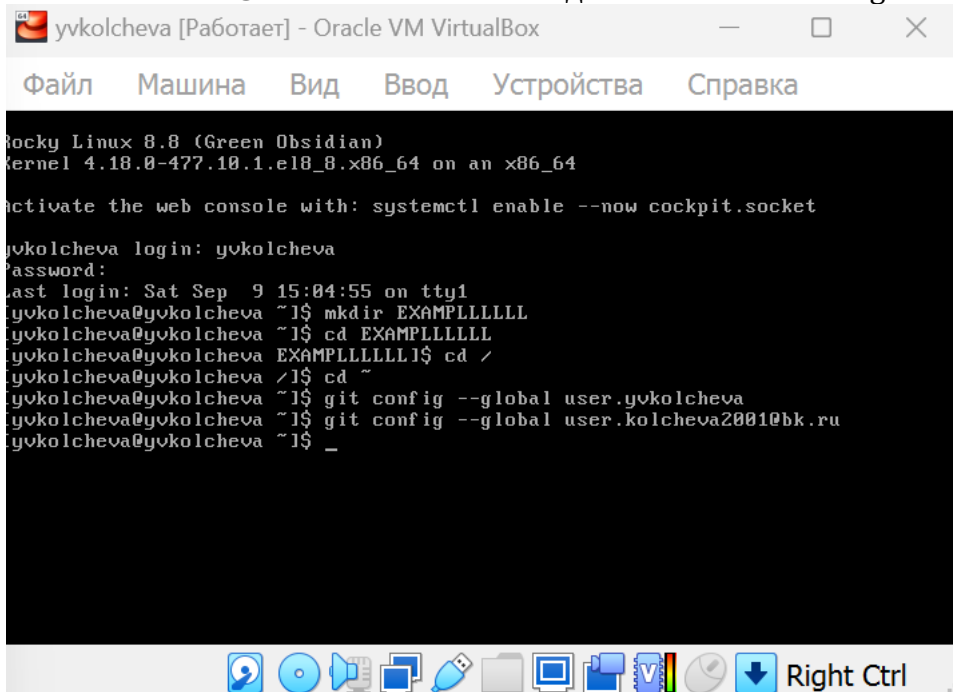


Figure 13: Создание учётной записи

2). Настраиваем систему контроля версий git. Синхронизируем учётную запись github с компьютером: `git config --global user.name "Имя Фамилия"` `git config --global user.email "work@mail"` После этого создаём новый ключ на github .



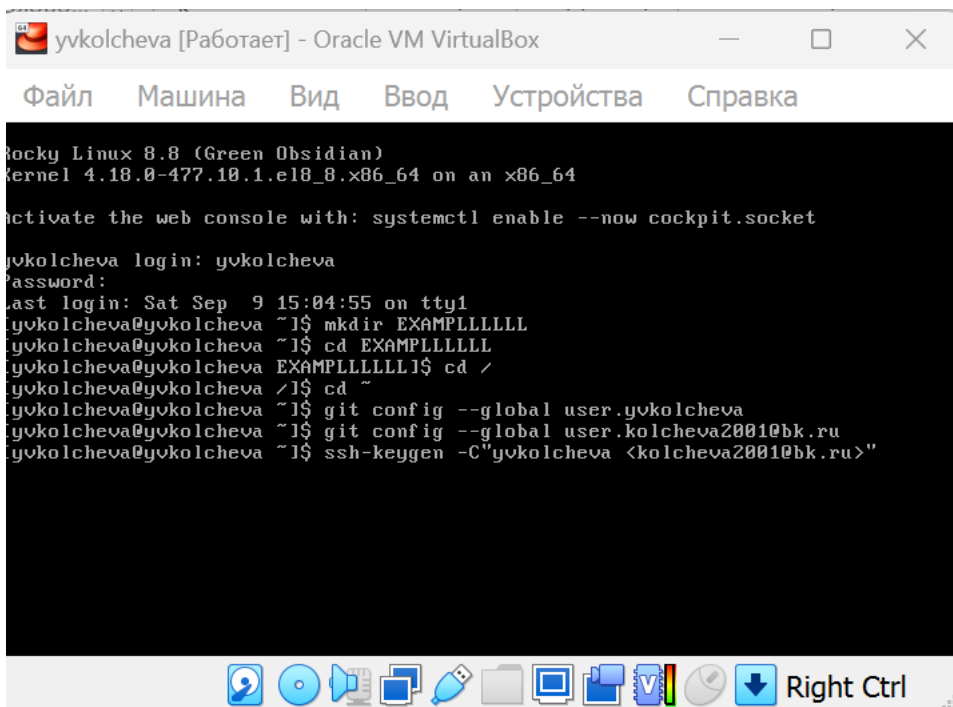


Figure 14: Создание нового ключа

3). Следующим шагом будет создание и подключение репозитория к github. В gethup заходим в «repository» и создаём новый репозиторий.

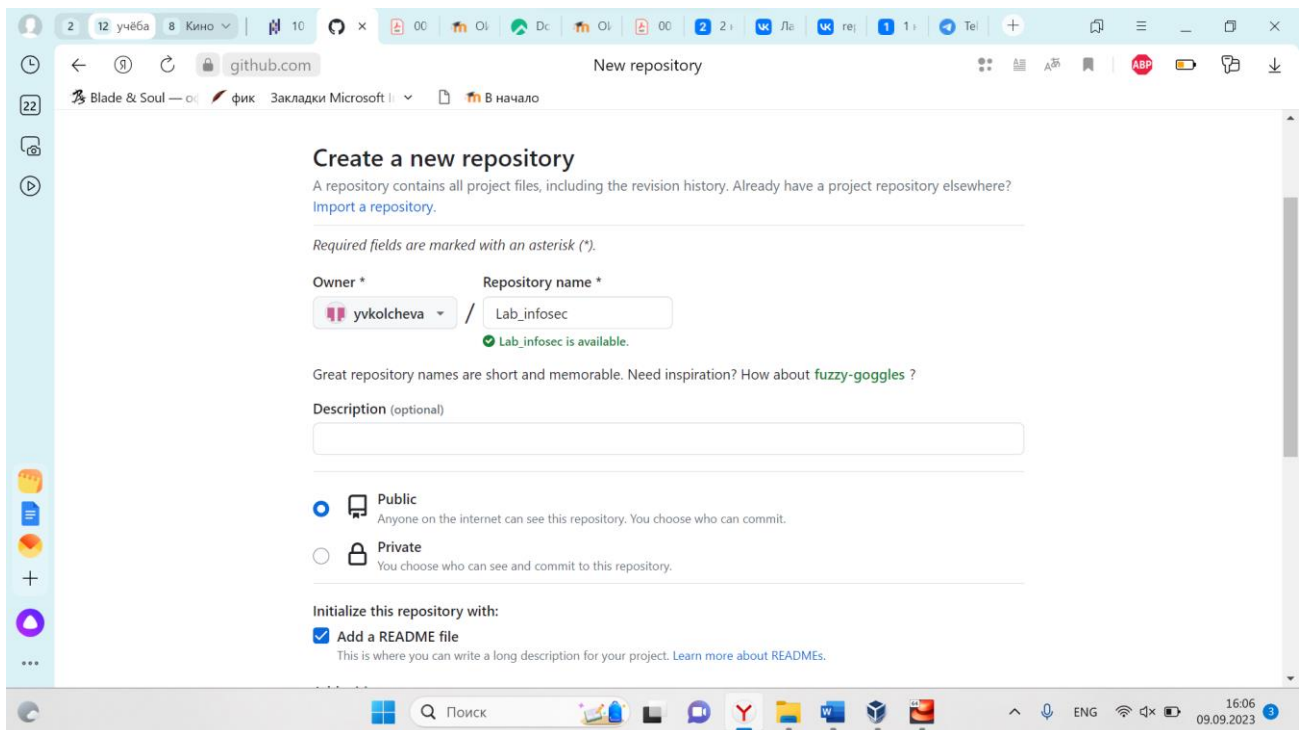


Figure 15: Создание репозитория

4). В лабораторной работе описан алгоритм создания структуры каталога через консоль. Но легче будет создать репозиторий в gethup и после этого работать с

каталогом и папками через консоль (перед этим необходимо скопировать ссылку на репозиторий в консоль, в формате https или ssh). Перед тем, как создавать файлы, заходим в наш репозиторий (алгоритм действий представлен на рис. 4.1):

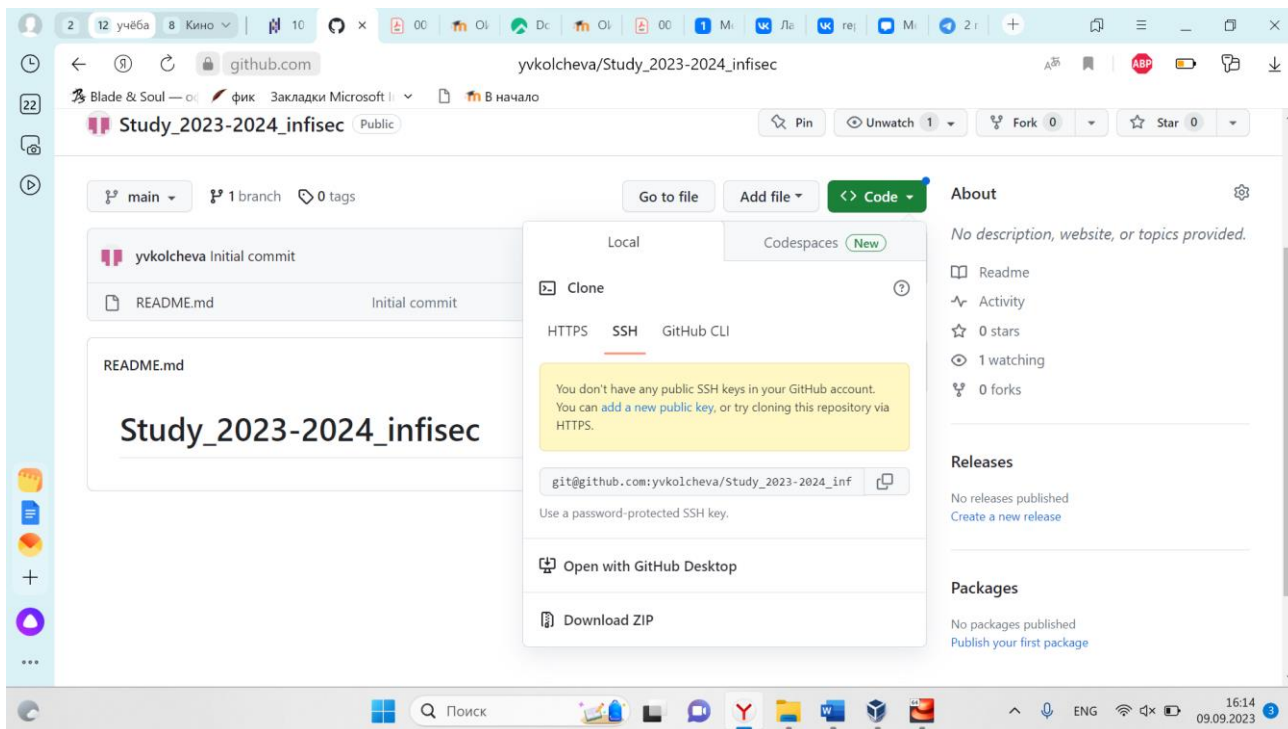


Figure 18: Создание репозитория

После этого можем уже создавать наши файлы:

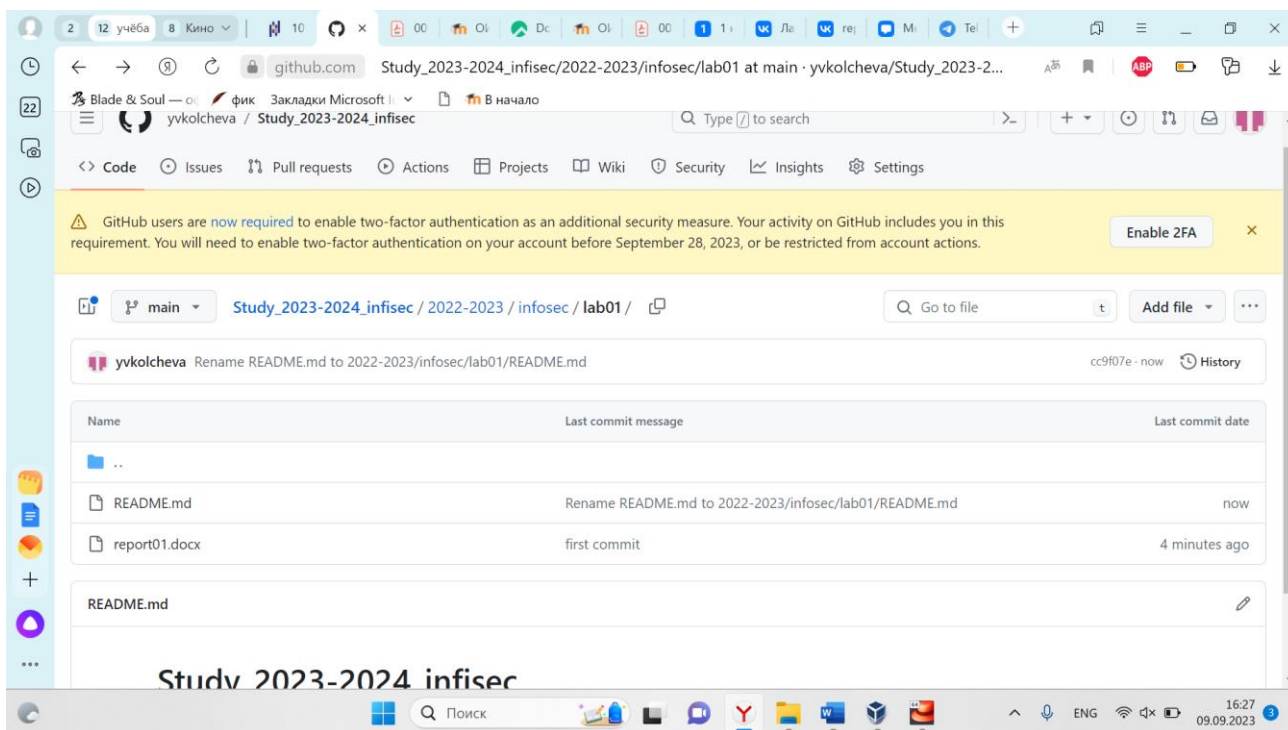


Figure 19: Создание файлов

5). Добавляем первый коммит и выкладываем на gethup. Для того, чтобы правильно разместить первый коммит, необходимо добавить команду `git add .`, после этого с помощью команды `git commit -am "first commit"` выкладываем коммит:

6). Сохраняем первый коммит, используя команду `git push`:



```
yvkolcheva@yvkolcheva ~]$ git push_
```

*Figure 21: Сохранение первого коммита*

## КОНТРОЛЬНЫЕ ВОПРОСЫ №2:

1). Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды `git` с различными опциями. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом.

2). 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 неполную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. 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.

3). Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Пример - Wikipedia. В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Пример — Bitcoin. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером.

4). Создадим локальный репозиторий. Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория: `git config --global user.name "Имя Фамилия"` `git config --global user.email "work@mail"` и настроив utf-8 в выводе сообщений `git config --global core.quotePath false` Для инициализации локального репозитория, расположенного, например, в каталоге `~/tutorial`, необходимо ввести в командной строке: `cd mkdir tutorial cd tutorial git init`

5). Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый): `ssh-keygen -C "Имя Фамилия work@mail"` Ключи сохраняются в каталоге `~/.ssh/`. Скопировав из локальной консоли ключ в буфер обмена `cat ~/.ssh/id_rsa.pub | xclip -sel clip` вставляем ключ в появившееся на сайте поле.

6). У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строки, а вторая — обеспечение удобства командной работы над кодом.

7). Основные команды git: Наиболее часто используемые команды git: – создание основного дерева репозитория: `git init` – получение обновлений (изменений) текущего дерева из центрального репозитория: `git pull` – отправка всех произведённых изменений локального дерева в центральный репозиторий: `git push` – просмотр списка изменённых файлов в текущей директории: `git status` – просмотр текущих изменений: `git diff` – сохранение текущих изменений: – добавить все изменённые и/или созданные файлы и/или каталоги: `git add .` – добавить конкретные изменённые и/или созданные файлы и/или каталоги: `git add имена_файлов` – удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): `git rm имена_файлов` – сохранение добавленных изменений: – сохранить все добавленные изменения и все изменённые файлы: `git commit -am 'Описание коммита'` – сохранить добавленные изменения с внесением комментария через встроенный редактор: `git commit` – создание новой ветки, базирующейся на текущей: `git checkout -b имя_ветки` – переключение на некоторую ветку: `git checkout имя_ветки` (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) – отправка изменений конкретной ветки в центральный репозиторий: `git push origin имя_ветки` – слияние ветки стекущим деревом: `git merge --no-ff имя_ветки` – удаление ветки: – удаление локальной уже слитой с основным деревом ветки: `git branch -d имя_ветки` – принудительное удаление локальной ветки: `git branch -D имя_ветки` – удаление ветки с центрального репозитория: `git push origin :имя_ветки`

8). Использование git при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий): `git add hello.txt` `git commit -am 'Новый файл'`

9). Проблемы, которые решают ветки git: • нужно постоянно создавать архивы с рабочим кодом • сложно “переключаться” между архивами • сложно перетаскивать изменения между архивами • легко что-то напутать или потерять

10). Во время работы над проектом так или иначе могут создаваться файлы, которые не требуются добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл `.gitignore` с помощью ервисов. Для этого сначала нужно получить списки имеющихся шаблонов: `curl -L -s https://www.gitignore.io/api/list` Затем скачать шаблон, например, для C и C++ `curl -L -s https://www.gitignore.io/api/c >> .gitignore` `curl -L -s https://www.gitignore.io/api/c++ >> .gitignore`

Часть 3. Создание Markdown файла В качестве подтверждения выполнения данной части лабораторной работы я прикрепляю отчёт в трёх форматах (docx, pdf, md). А также презентации в двух форматах (pdf, md).

## Выводы

Установила VirtualBox, изучила её работу. Изучила идеологию и научилась применять средства контроля версий. Научилась работать с Markdown-файлами.