Лабораторная работа №1

Дисциплина: Основы информационной безопасности

Аветисян Давид Артурович

Содержание

List of Figures

List of Tables

1 Цель работы

Часть 1: установка на виртуальную машину системы Rocky Linux и настройка необходимых сервисов. Часть 2: освоение git и применение средств контроля версий. Часть 3: оформление отчётов с помощью языка разметки Markdown.

2 Задание

Часть 1 1.Установка и настройка Виртуальной машины VMware (ОС Linux)

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

3 Выполнение лабораторной работы

Часть 1. Установка и настройка Виртуальной машины VMware

1) В качестве виртуальной машины я решил выбрать VMware, которая уже была установлена у меня на компьютере.

2) Создаём виртуальную машину. Для этого нажимаем на “Создать”. Имя нашей виртуальной машины - daavetisyan. Выбираем наш образ диска Rocky Linux. Чтобы он распознал его как образ. Автоматически выбирается Linux версии RedHat.

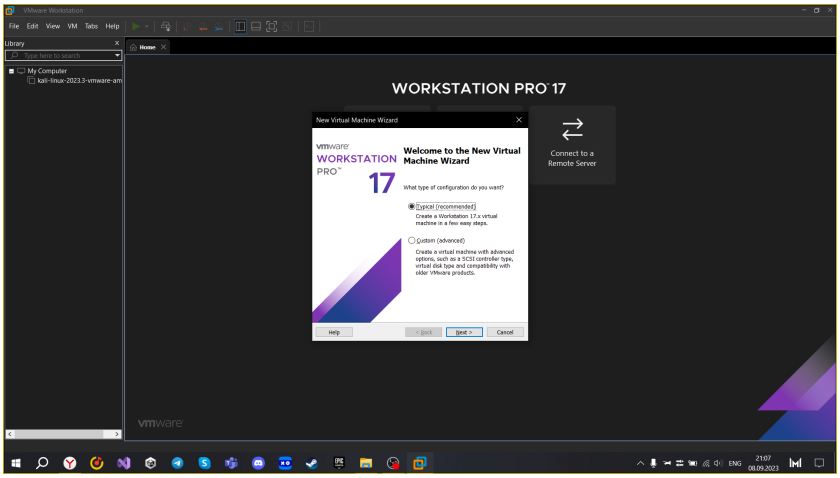

Figure 3.1: Создание виртуальной машины

3) Выделяем 20 ГБ памяти на виртуальную машину, думаю, этого достаточно.

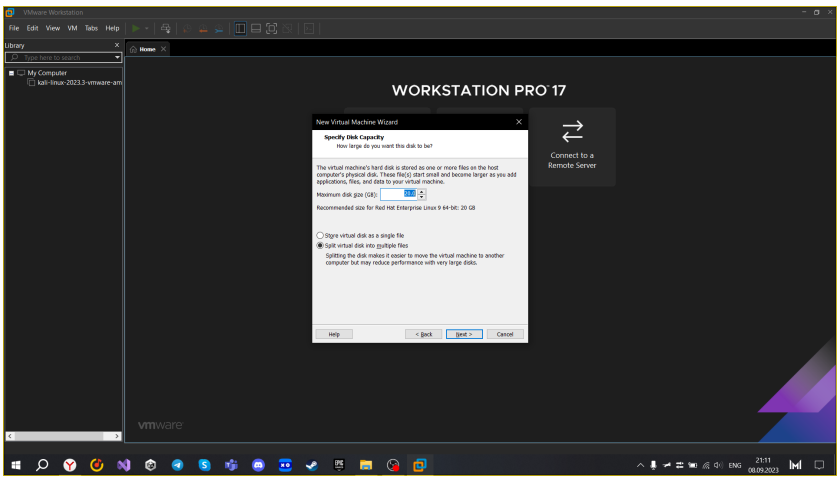

Figure 3.2: Основная память

4) Виртуальной машине требуется ОЗУ. Выделяем 2048 МБ, что является четвертью основного ОЗУ. Также выделяем ей 2 ЦП.

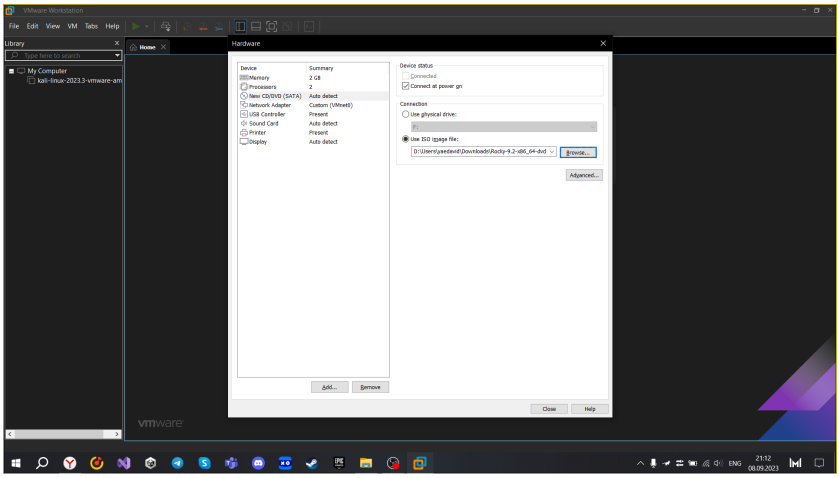

Figure 3.3: Память и процессоры

5) Включаем виртуальную машину, нас встречает выбор языка.

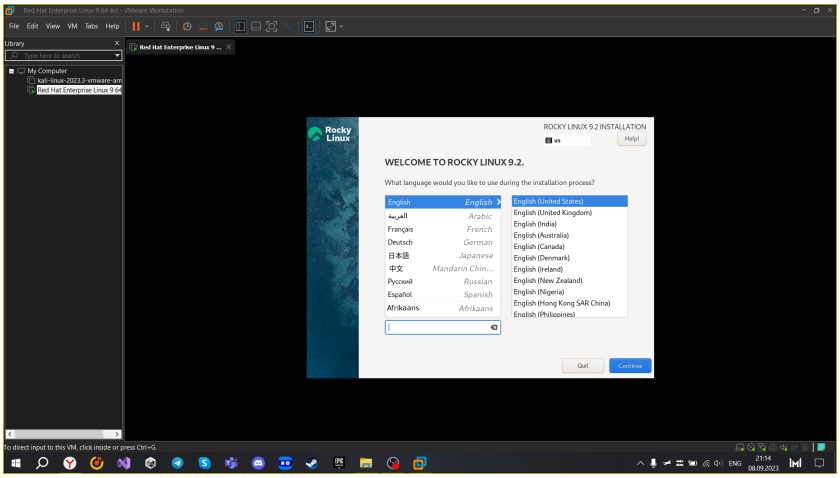

Figure 3.4: Выбор языка

6) О тключаем KDUMP. Настраиваем сеть и имя узла. Базовое окружение выбираем Сервер с GUI. В дополнительном окружении выбираем средства разработки.

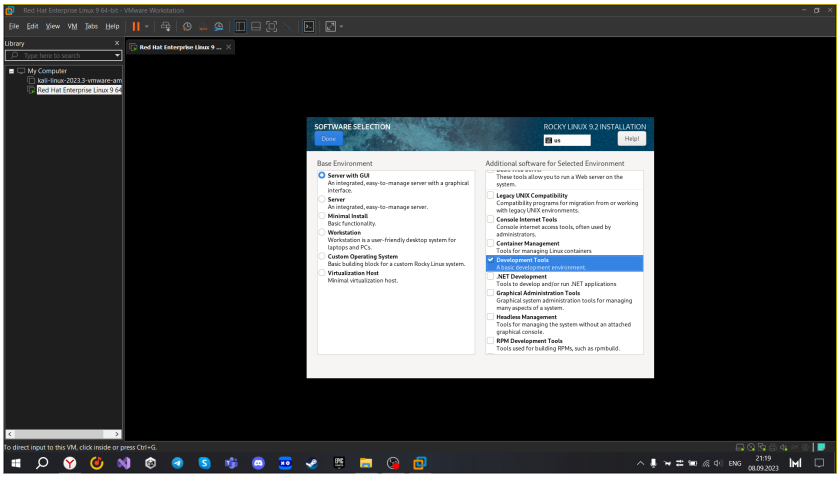

Figure 3.5: Выбор программ

9) Устанавливаем пароль для root пользователя.

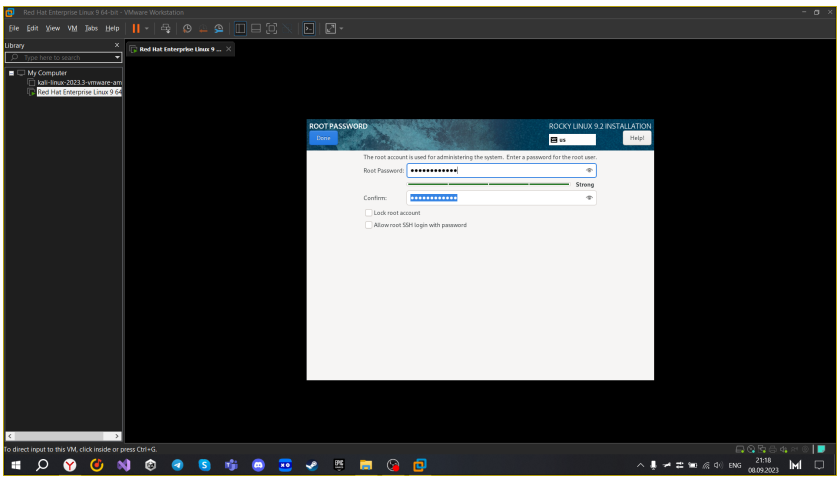

Figure 3.6: Пароль

10) В выборе места установки ставим наш диск.

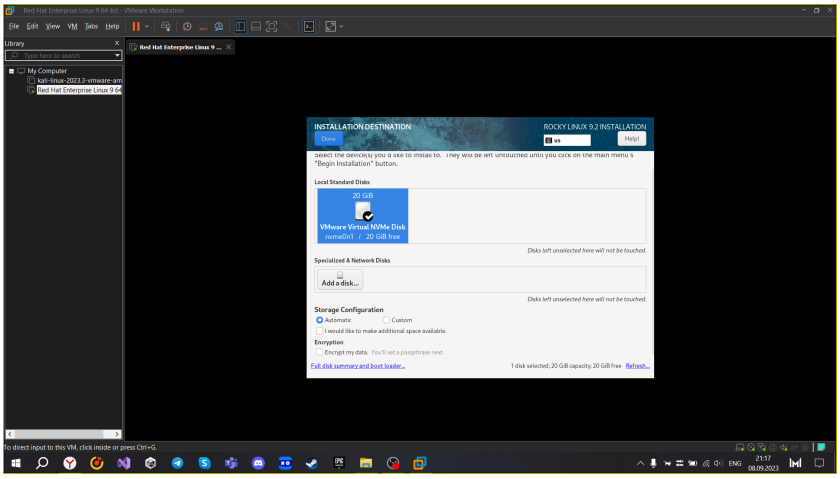

Figure 3.7: Место установки

11) Установка закончена. Перезагружаем.

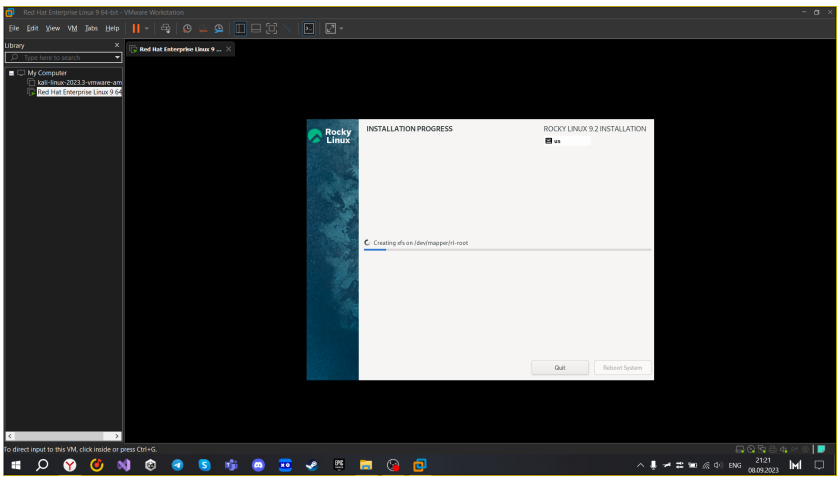

Figure 3.8: Перезагрузка

12) Нас встречает система Rocky Linux. Установка завершена! Осталось подключить образ диска дополнений гостевой ОС.

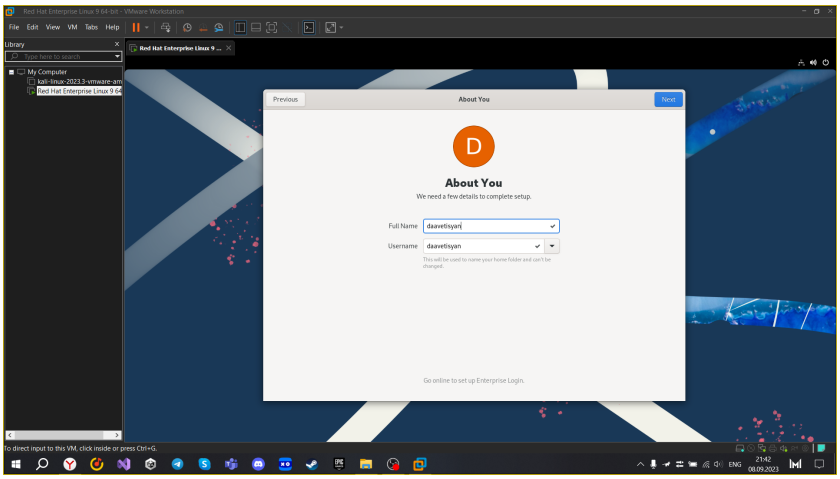

Figure 3.9: Rocky Linux

ДОМАШНЕЕ ЗАДАНИЕ №1:

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

Можно использовать поиск с помощью grep: dmesg | grep -i “то, что ищем” Команда grep используется для поиска данных. а. Версия ядра Linux (Linux version).

b. Частота процессора (Detected Mhz processor). c. Модель процессора (CPU0) d. Объем доступной оперативной памяти (Memory available) e. Тип обнаруженного гипервизора (Hypervisor detected). f. Тип файловой системы корневого раздела. Последовательность монтирования файловых систем

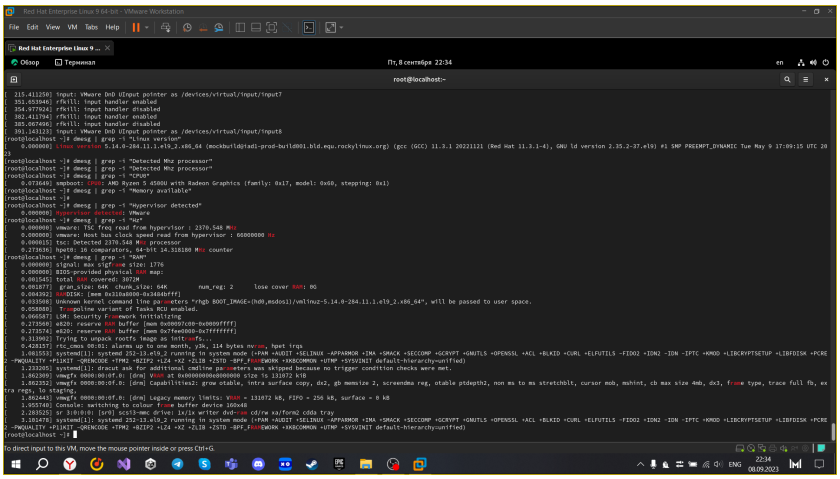

Figure 3.10: dmesg

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: a.Для получения справки используется ключ –help или команда man. Например, ls –help или man ls. b.Для перемещения по файловой системе используется команда cd. Например cd ~. c.Для просмотра содержимого каталога используется команда ls. Например ls ~/work. d.Для определения объёма каталога используется команда du. e.Для создания каталогов используется mkdir, для удаления пустых каталогов используется rmdir. Для создания файлов используется touch, для удаления файлов и каталог используется rm. f.Для задания прав используется команда chmod. Например, chmod u-w test.txt. g.Для просмотра истории команд используется команда history.

3) Файловая система — часть ОС, которая обеспечивает чтение и запись файлов на дисковых носителях информации. a.Ext2 — расширенная файловая система. Данные сначала кэшируются и только потом записываются на диск. b.Ext3 и Ext4 — журналируемые файловые системы. Осуществляется хранение в виде журнала со списком изменений, что помогает сохранить целостность при сбоях. c.XFS — высокопроизводительная журналируемая файловая система, рассчитанная для работы на дисках большого объёма.
4) Для просмотра подмонтированных в ОС файловых систем необходимо использовать команду findmht.
5) Для удаления зависшего процесса используется команда kill PID или killall название.

Часть 2. Работа с GitHub

1) Первым делом, мы регистрируемся на сайте github.com. Так как мы не в первый раз имеем дело с github, то аккаунт уже готов.

2) Первоначальная настройка для работы с github.

```
Инициализирован пустой репозиторий Git в /home/daavetisyan/work/study/2023-2024/infosec/.git/
[daavetisyan@daavetisyan infosec]$ git branch
[daavetisyan@daavetisyan infosec]$ git config --global user.name "yaedavid"
[daavetisyan@daavetisyan infosec]$ git config --global user.email "yaedavid@ya.ru"
[daavetisyan@daavetisyan infosec]$ git config --global core.quotepath false
[daavetisyan@daavetisyan infosec]$ git config --global init.defaultBranch master
[daavetisyan@daavetisyan infosec]$ git config --global core.autocrlf input
[daavetisyan@daavetisyan infosec]$ git config --global core.safecrlf warn
[daavetisyan@daavetisyan infosec]$
```

Figure 3.11: Настройка

3) Генерация GPG ключа.

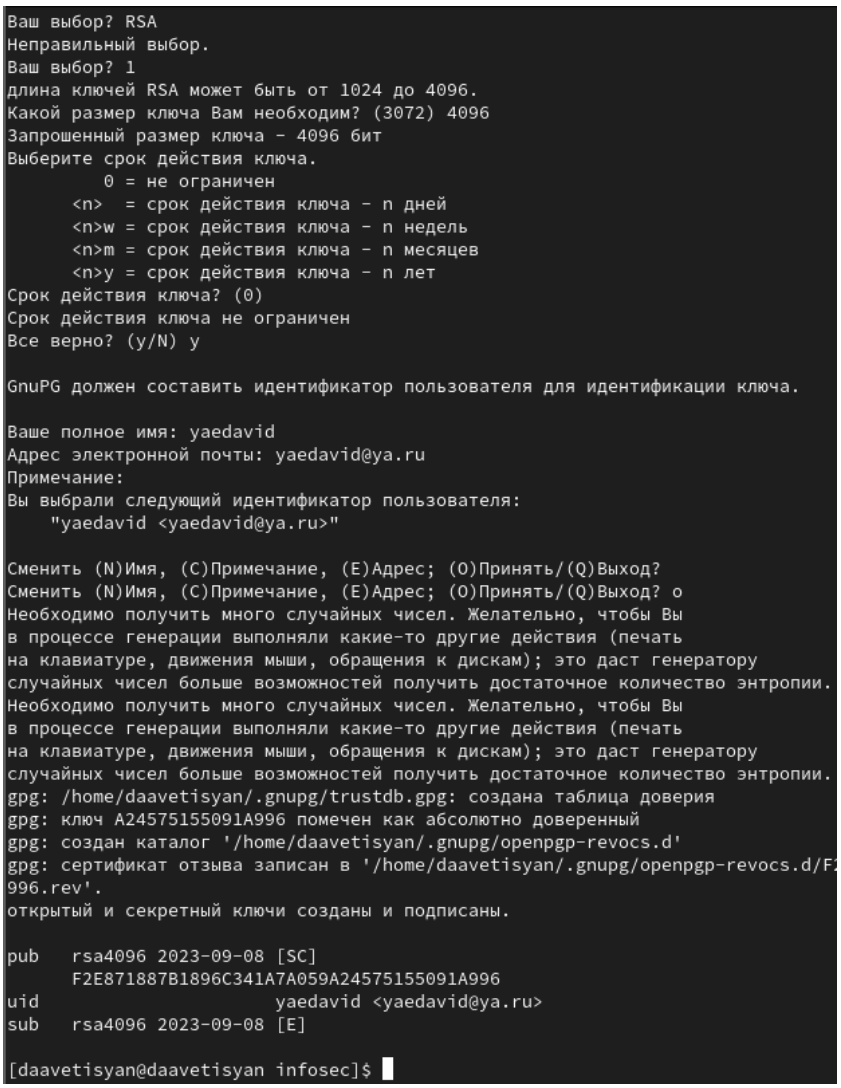

Figure 3.12: Ключ

4) Переходим на сайте в настройки и в GPG Keys. Копируем наш GPG ключ, предварительно выведя его в консоль. Также нужно было написать git init в терминале для создания основного дерева репозитория.

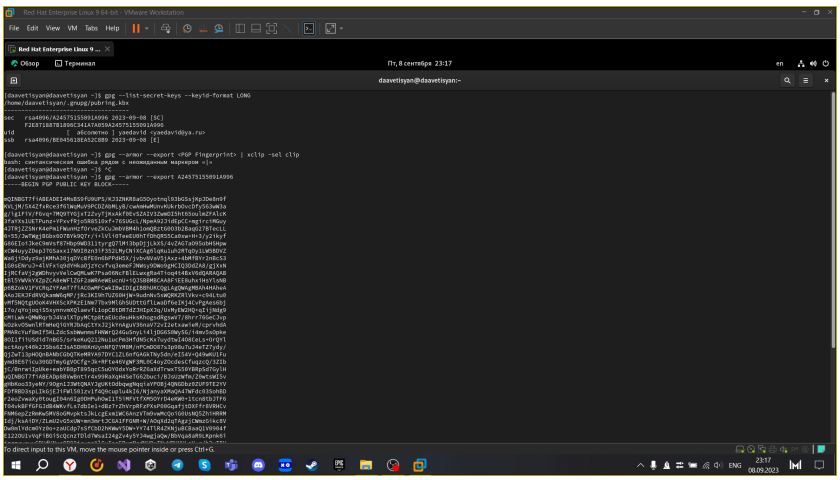

Figure 3.13: Вывод ключа в терминале

5) Ключ отобразился на сайте.

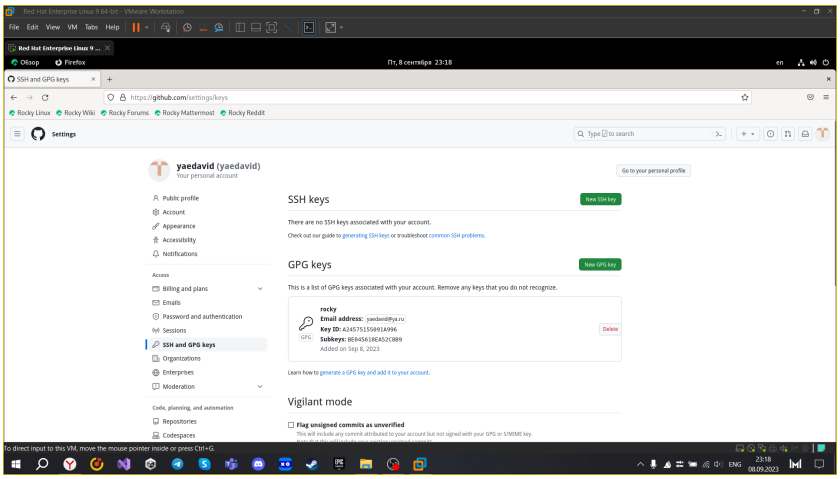

Figure 3.14: Ключ

6) Клонируем наш репозиторий work из 2021, когда мы работали с ОС. Создаём рабочее пространство для наших лабораторных работ, как сказано в документации.

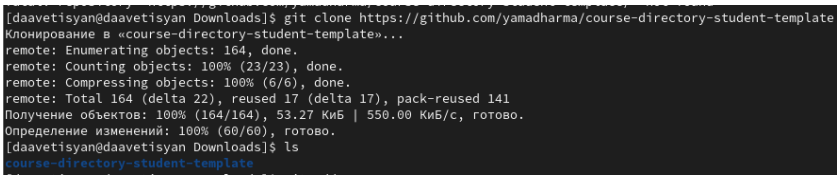

Figure 3.15: Рабочее пространство

7) Пишем git add. После этого делаем первый коммит за 2 года. Отправляем изменения на github.

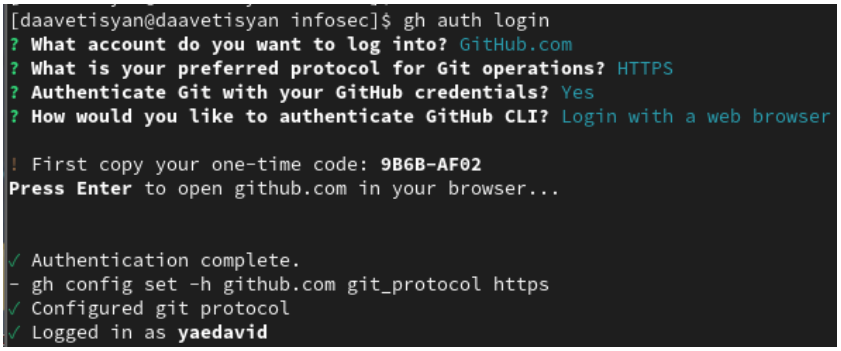

Figure 3.16: Коммит

Вторая часть закончена. Мы связали наш репозиторий с репозиторием на github.com.

КОНТРОЛЬНЫЕ ВОПРОСЫ №2:

1) Система контроля версий Git представляетсобой набор программ командной строки.Доступ к ним можно получить изтерминала посредством ввода командыgitc различ-ными 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 определённыхкоманд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляютсяиз 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,чтопозволяет 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. Крометого, 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 сообщенийgit: git config –global 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 репозиториев необходимо сгенерировать пару ключей (приватный и открытый): ssh-

keygen -C”Имя Фамилия work@mail” Ключи сохраняться 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 строчки, 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 втекущей директории:git status–просмотртекущих изменения:git diff–сохранениетекущих 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 (приэтомфайл и/илик аталог 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 натекущей: 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 цен-

трального репозитория: git push origin :имя_ветки

8) Использования 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 проектомтак или иначе могутсоздаваться файлы,которые нетребуется добавлять в последствии в репозиторий. Например, временные файлы, со-здаваемые редакторами,или объектные файлы, создаваемые компиляторами.Можно прописать шаблоны игнорируемых при добавлении в репозиторийтипов файлов в файл.gitignore с помощьюс ервисов. Для этого сначала нужно получить списоки меющихся 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

1) Для того, чтобы из Markdown получить PDF и DOCX файлы, требуется скачать pandoc и pandoc-crossref. Это непростая задача, пришлось скачивать их с github.com старой версии 2.14. Переместить исполняемые файлы в папку ~/bin. Сперва скачаем zip-файлы из гитхабов jgm и lierdakil. Смотрим ветки и выбираем всё для pandoc 2.14.

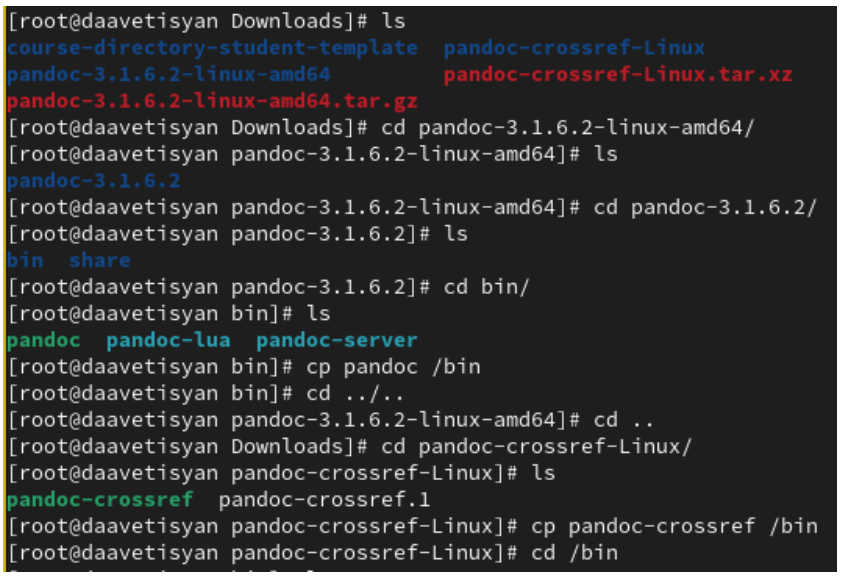

Figure 3.17: pandoc

2) Таким образом, как на скриншоте, мы перемещаем в корневую папку pandoc. Аналогично с pandoc-crossref.

3) В консольной строке пришем make и получаем наши файлы. Примечание: компиляцию файлов я делал в Ubuntu, так как в Rocky это делается немножко криво. Нужно не забыть написать команду sudo apt-get install texlive-full, чтобы все нужные шрифты были готовы.

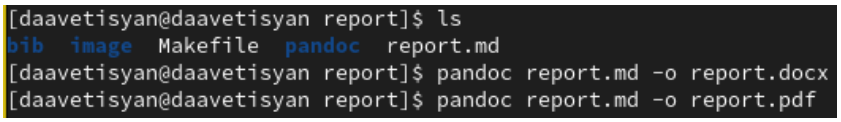

Figure 3.18: Компиляция файлов

4 Выводы

Я установил VMware, поставил на виртуальную машину Rocky Linux. Разобрался с системой контроля версий и github. Скомпилировал файлы из формата md в pdf и docx.