Лабораторная работа №1 и №2

Установка и конфигурация операционной системы на виртуальную машину и управление версиями

Кекишева Анасатасия Дмитриевна

Содержание

# Цель работы

* Приобретение практических навыков установки операционной системы на виртуальную машину, настройки минимально необходимых для дальнейшей работы сервисов.
* Изучение идеологии и применение средств контроля версий.
* Освоение работы с git.

# Задание

Для лабораторной №1:

Получить следующую информацию: - Версия ядра Linux (Linux version). - Частота процессора (Detected Mhz processor). - Модель процессора (CPU0). - Объем доступной оперативной памяти (Memory available). - Тип обнаруженного гипервизора (Hypervisor detected). - Тип файловой системы корневого раздела.

Для лабораторной №2:

* Создать базовую конфигурацию для работы с git.
* Создать ключ SSH .
* Создать ключ PGP .
* Настроить подписи git.
* Зарегистрироваться на Github .
* Создать локальный каталог для выполнения заданий по предмету.

# Теоретическое введение

Опишем теоретические аспекты, связанные с выполнением работы.

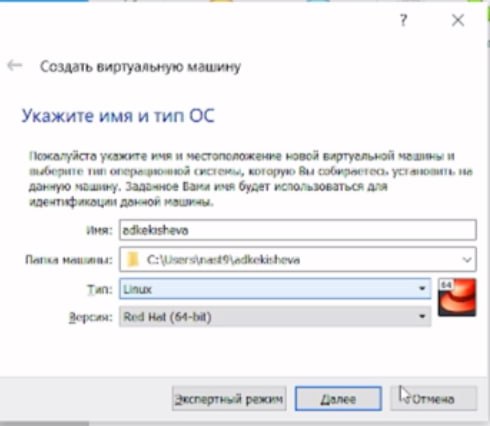
Системы контроля версий (Version Control System, VCS) применяются при работе несколь- ких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля 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 боль- шинства 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 (авто- матически 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 нескольки- ми 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 распределён- ных — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

C помощью встроенной утилиты «gh» можно создавать и разбирать пул-реквесты, включая работу в команде, заводить и просматривать сообщения об ошибках (issue), клонировать репозитории, просматривать различия в коде, выполнять рецензирование изменений и управлять формированием релизов на GitHub.

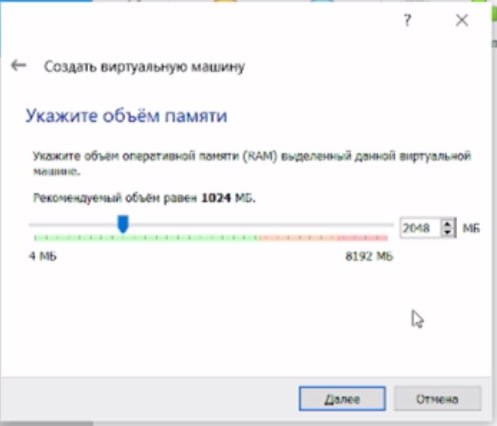
# Выполнение лабораторной работы

1. Скачала виртуальную машину и дистрибутив [Centos](https://%20www.centos.org/)
2. Создала новую виртуальную машину, указала имя виртуальной машины – adkekisheva. Выбрала тип операционной системы — Linux, RedHat (рис. @fig:001).



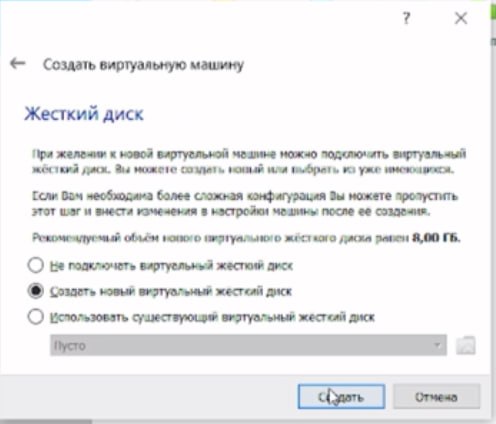
Имя и тип ОС

1. Указала размер основной памяти виртуальной машины — 2048 МБ (рис. @fig:002).

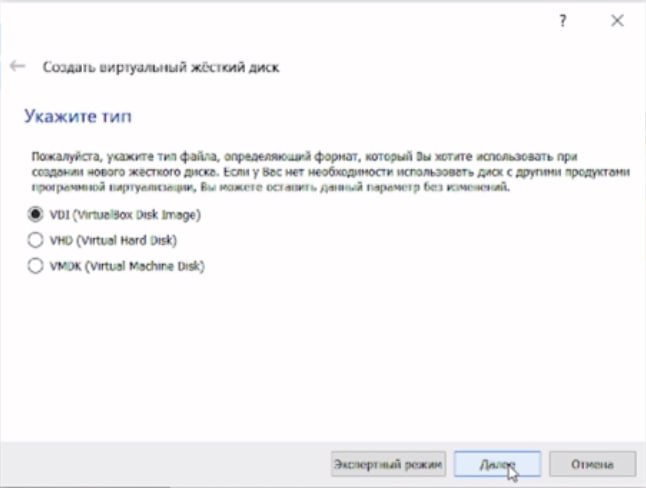


Объём памяти

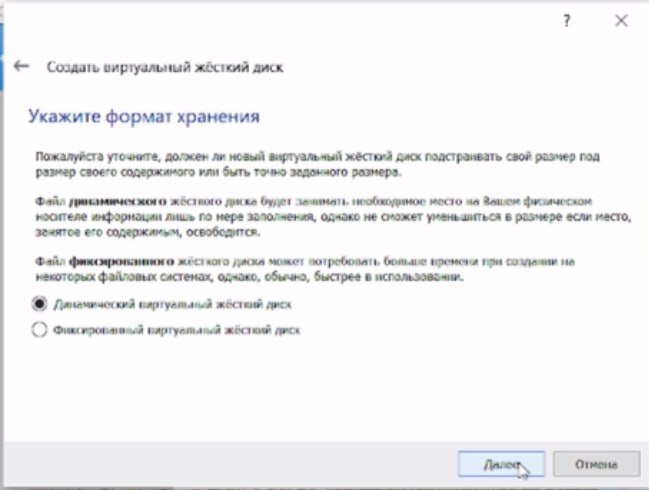
1. Создала новый вируальный диск (рис. @fig:003), задала конфигурацию жёсткого диска — загрузочный VDI (рис. @fig:004), выбрала динамический виртуальный диск (рис. @fig:005).



Создание нового жёского диска

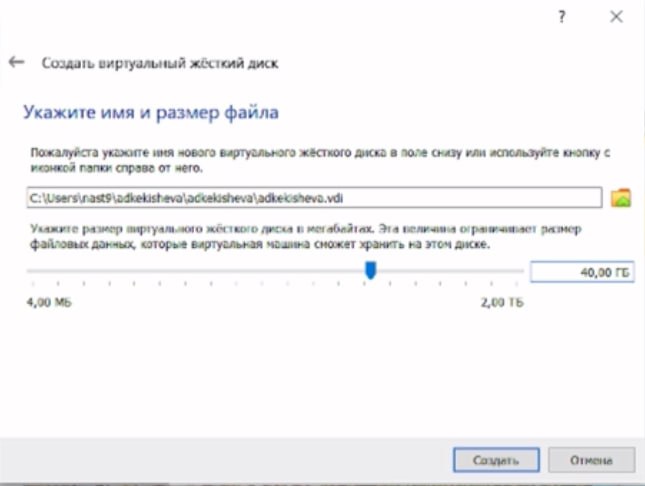


Тип диска - VDI



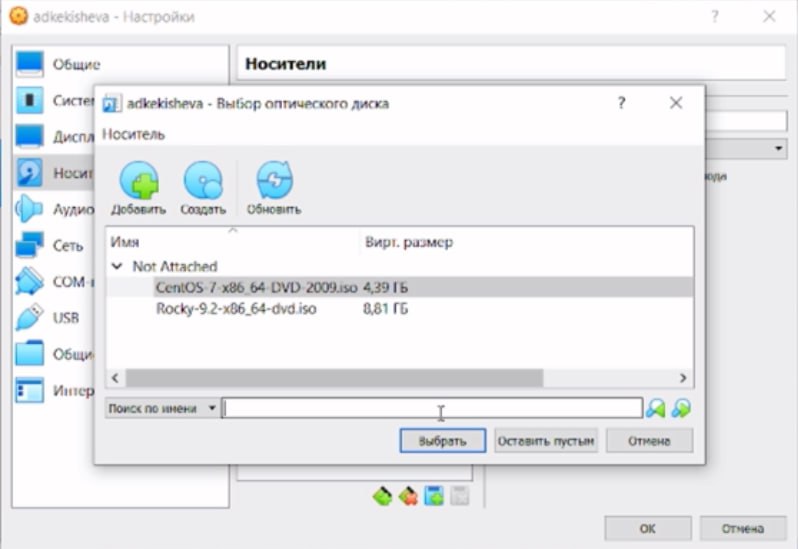
Формат хранения - динамический

1. Задала размер диска — 40 ГБ и его расположение (рис. @fig:006).



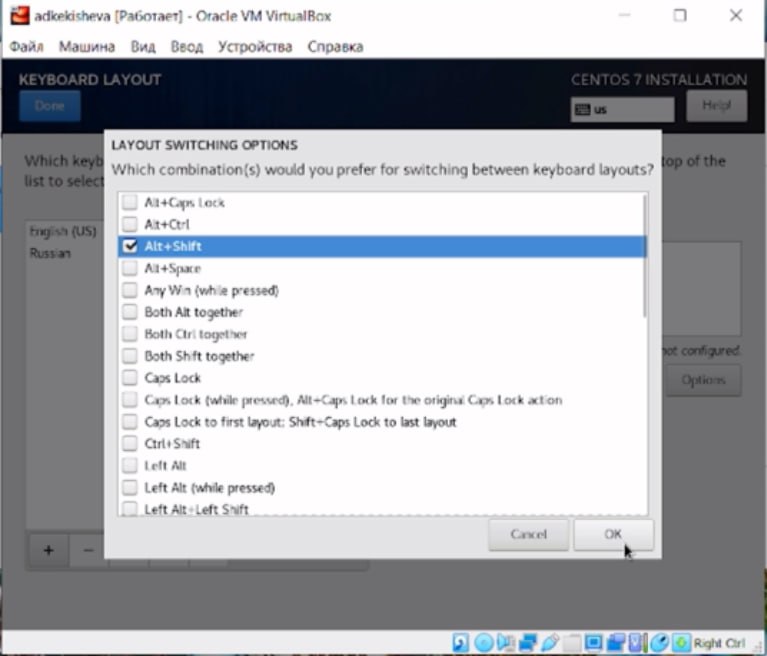
Определение размера диска

1. Добавила новый привод оптических дисков и выбрала образ операционной системы (рис. @fig:007).



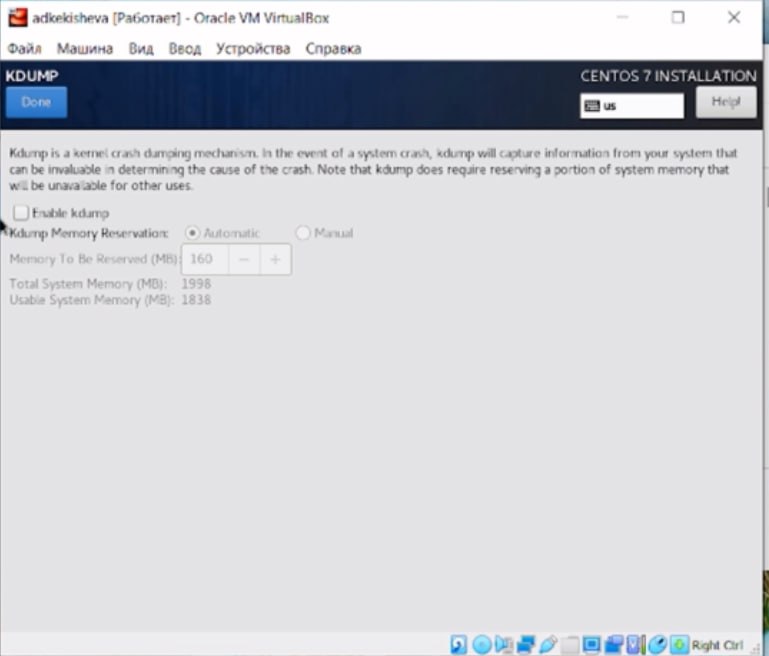
Добавление оптического диска и выбор образа CentOS

1. Запустила виртуальную машину, скорректировала часовой пояс, раскладку клавиатуры (рис. @fig:008).



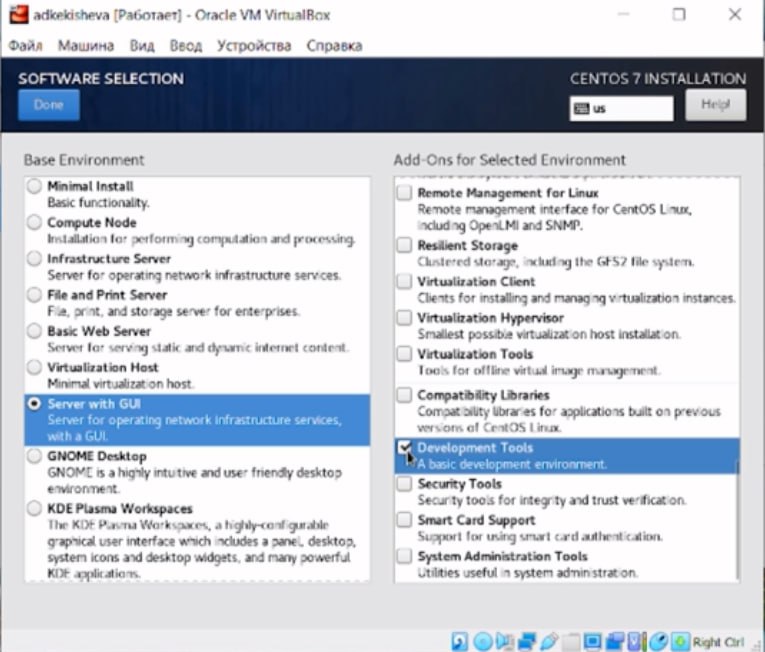
Настройка клавиатуры

1. Отключила KDUMP (рис. @fig:009).



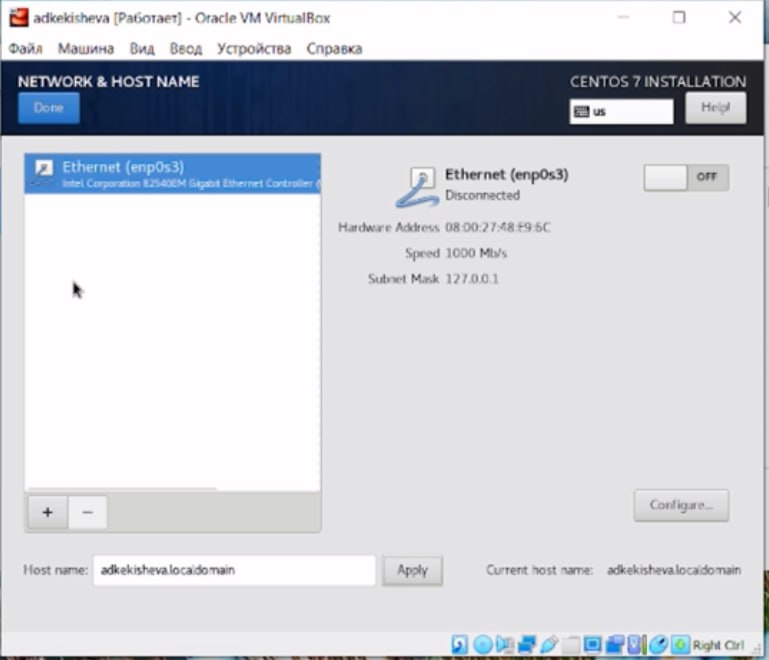
Отключение KDUMP

1. Указала в качестве базового окружения Server with GUI , а в качестве дополнения — Development Tools (рис. @fig:010).



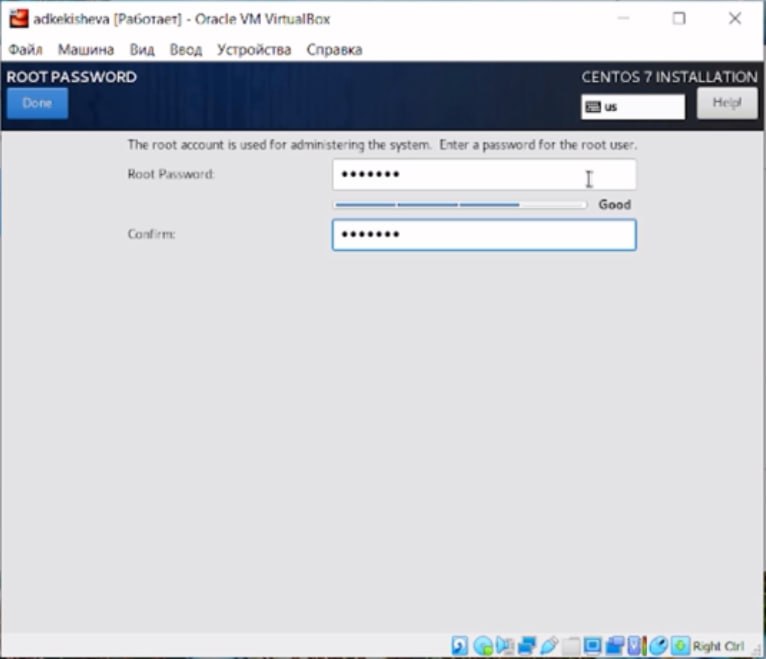
Настройка базового окружения

1. Включила сетевое соединение и в качестве имени узла указала adkekisheva.localdomain (рис. @fig:011).

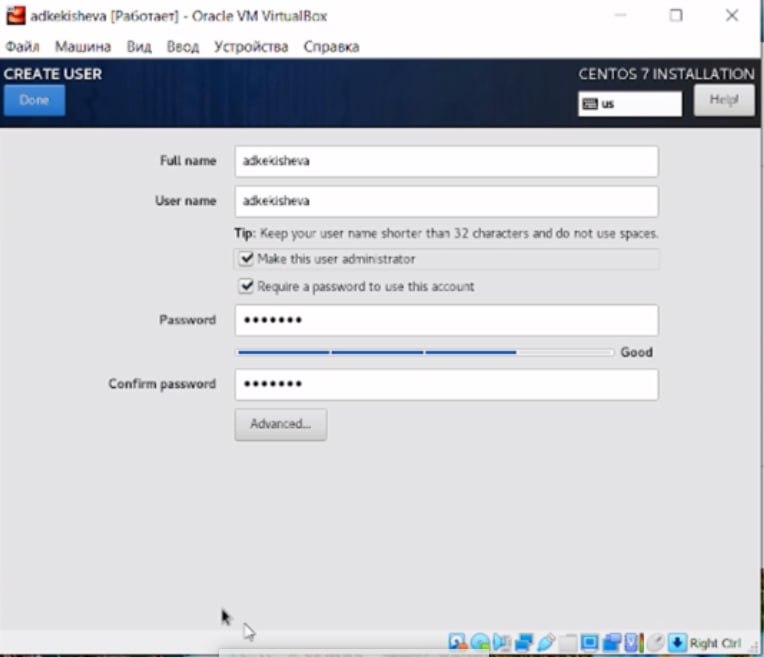


Настройка сетевого узла

1. Установила пароль для root (рис. @fig:012) и пользователя с правами администратора (рис. @fig:013).

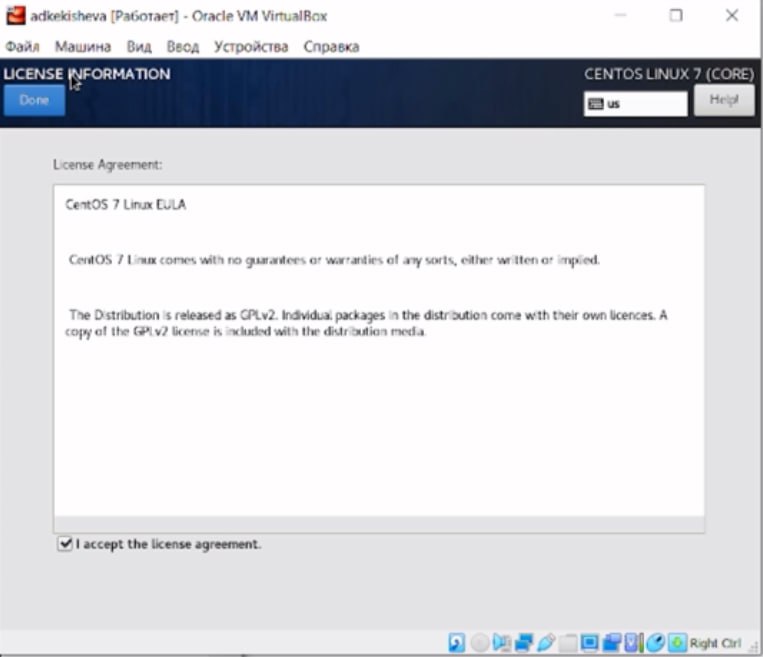


Пароль для root



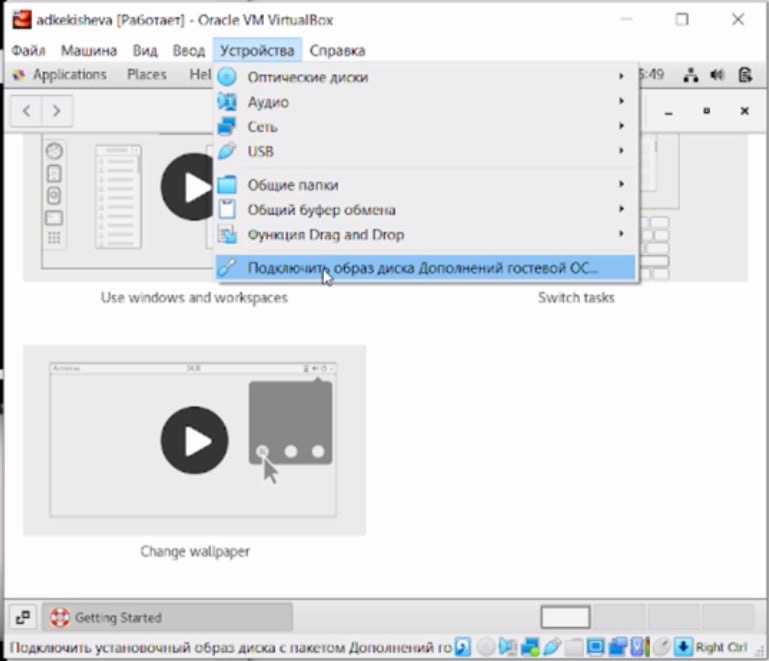
Создание пользователя с правами администратора

1. После завершения установки операционной системы перезапустила виртуальную машину и приняла условия лицензии (рис. @fig:011).



Принятие лиценционного соглашения

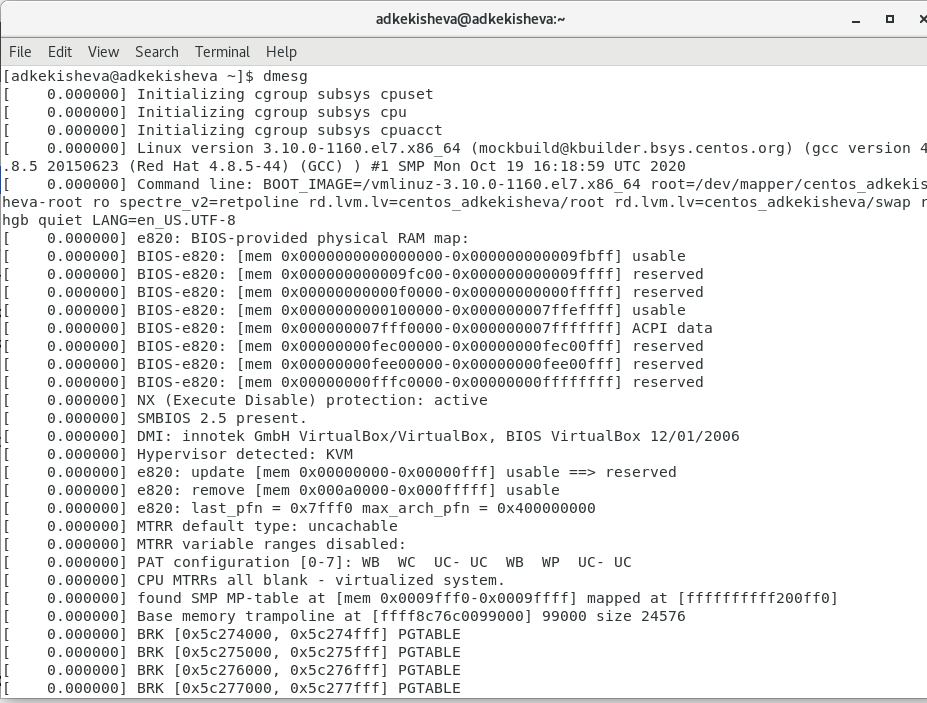
1. Вошла в ОС под заданной учётной записью и установила драйверы (рис. @fig:015).



Установка драйверов

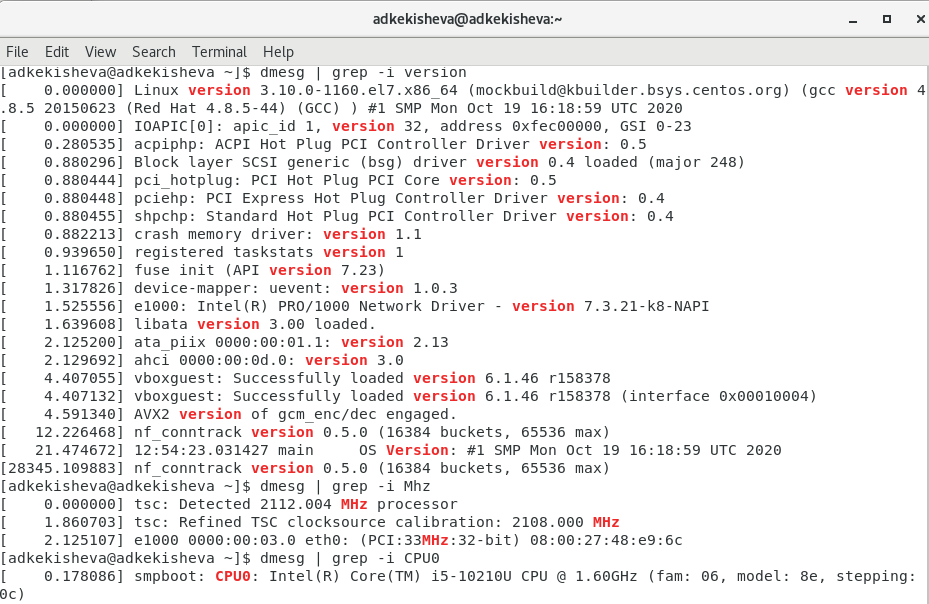
Далее перешла к аналиизу последовательности загрузки системы.

1. Выполнила команду dmesg в консоли (рис. @fig:016) и получилось, что загрузка происходит в следующей последовательности: 1 - загрузка и инициализация ядра; 2 -обнаружение и конфигурирование устройств; 3 - создание процессов ядра; 4 - выполнение сценариев запуска; 5 - работа в многопользовательском режиме.



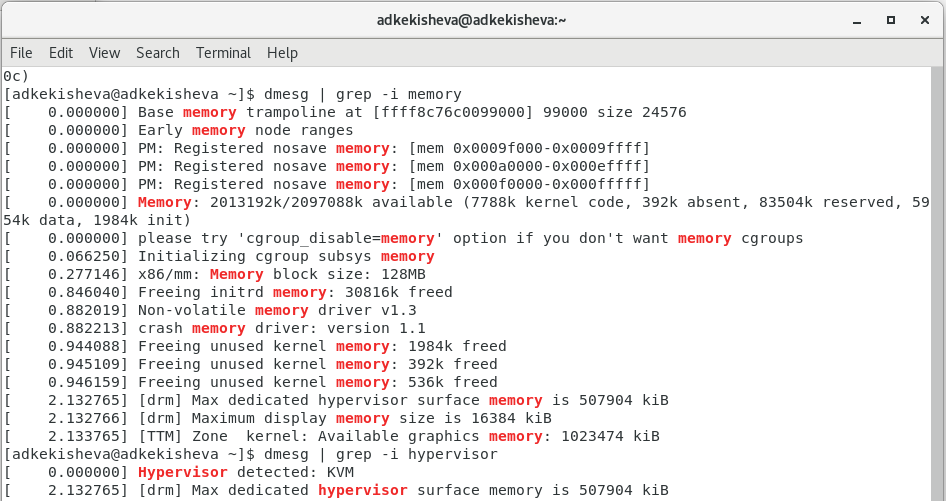
Выполнение команды gmesg

1. С помощью команды dmesg | grep -i version получила версию ядра Linux, которая равна 3.10.0 (рис. @fig:017).
2. С помощью команды dmesg | grep -i Mhz получила частоту процессора, равную 2112.004 Mhz (рис. @fig:017).
3. С помощью команды dmesg | grep -i CPU получила модель процессора (CPU0) (рис. @fig:017).



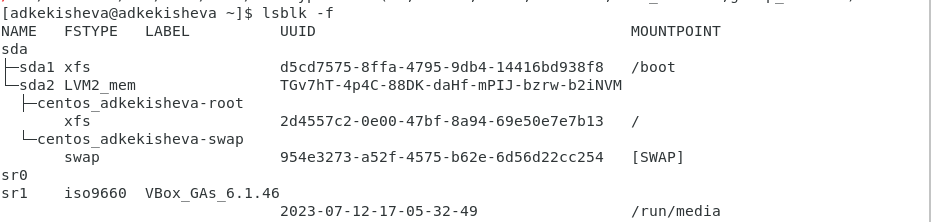
Нахождение версии ядра Linux, частоты и модели

1. С помощью команды dmesg | grep -i memory узнала, что объем доступной оперативной памяти составляет 2013192k (рис. @fig:018).
2. С помощью команды dmesg | grep -i hypervisor нашла тип обнаруженного гипервизора, это – KVM (рис. @fig:018).



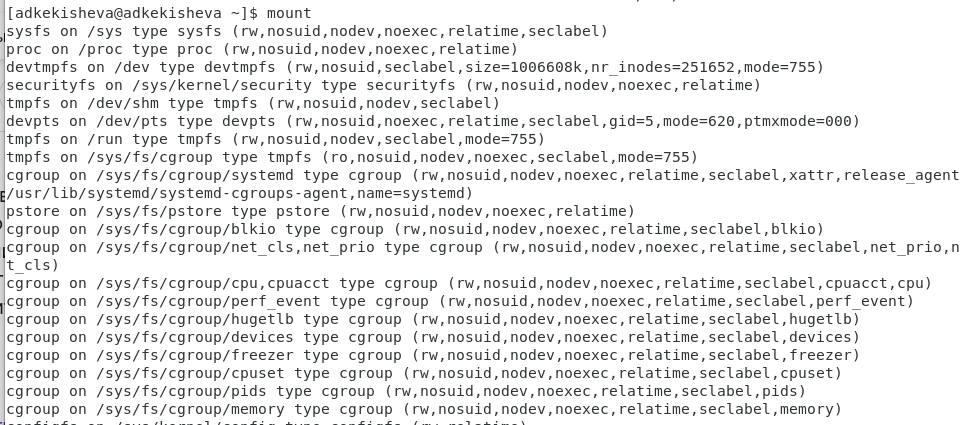
Нахождение оперативной памяти и тип гипервизора

1. С помощью команды lsblk -f узнала тип файловой системы корневого раздела – xfs (рис. @fig:019).



Иерархия файловых систем

1. Последовательность монтирования файловых систем посмотрела командой mount (рис. @fig:020).



Выполнение команды mount

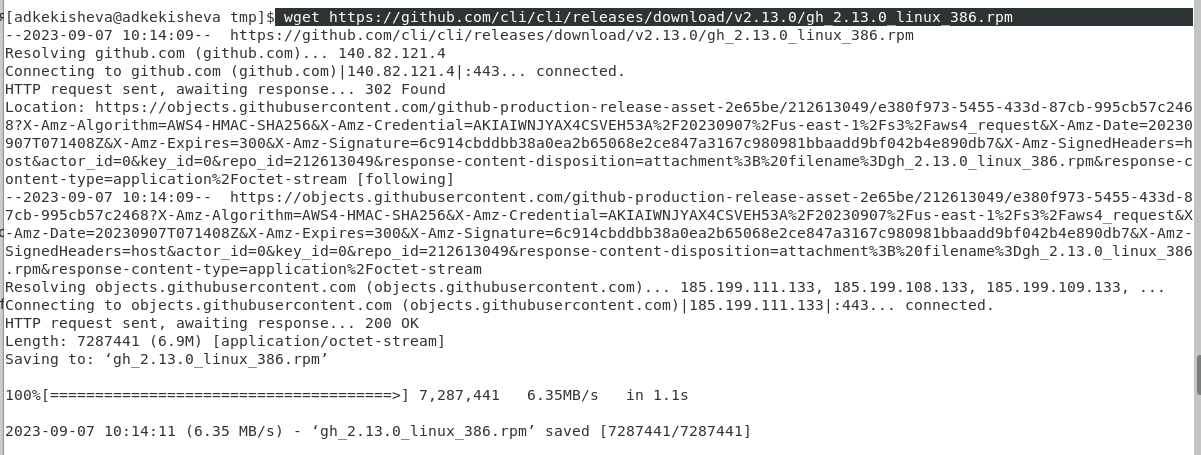
**На этом часть лабораторной выполнена, переходим ко второй части.**

1. Для начала установила программное обеспечение git-flow (рис. @fig:021).

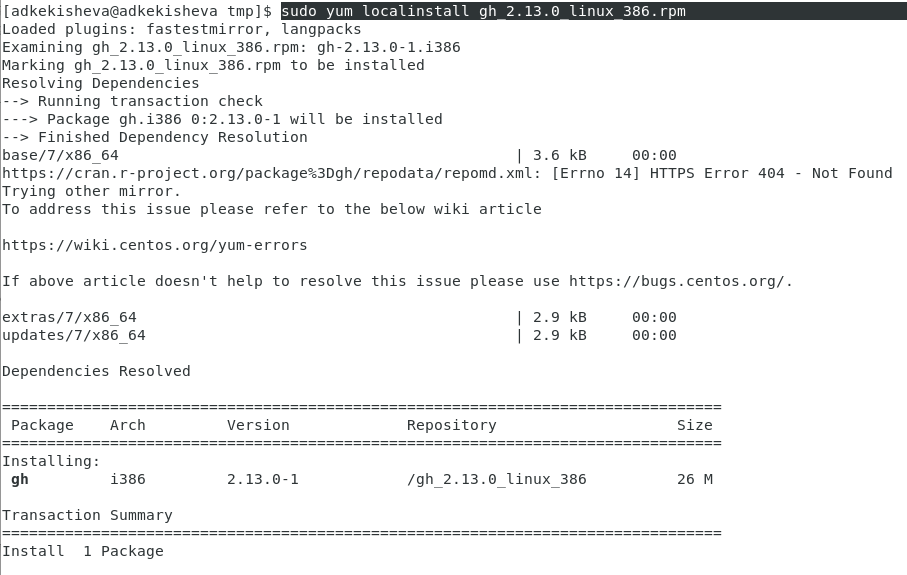


Установка git-flow

1. Установила gh, однако обычной командой он у меня не скачивался, поэтому скачала, через пакет (рис. @fig:022 и @fig:023).

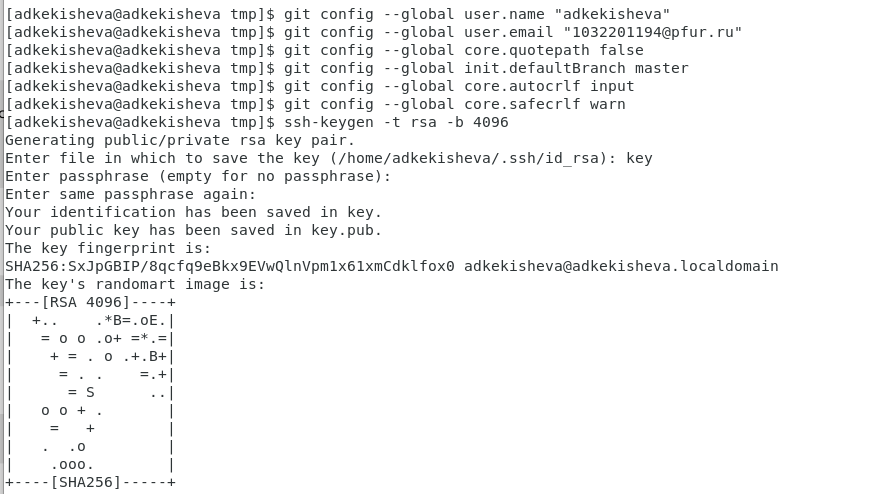


Скачивание пакетв gh



Установка пакета gh

1. Задала имя и email для моего репозитория, настроила utf-8 в выводе сообщений git, задала имя master для начальной ветки, параметры autocrlf и safecrlf, что означает, что файлы проверяются как есть, в при коммите конфертируются из CRLF в LF для записи техстовых файлов. Также создала ключ SSH по алгоритму RSA с размером 4096 бит (рис. @fig:024).



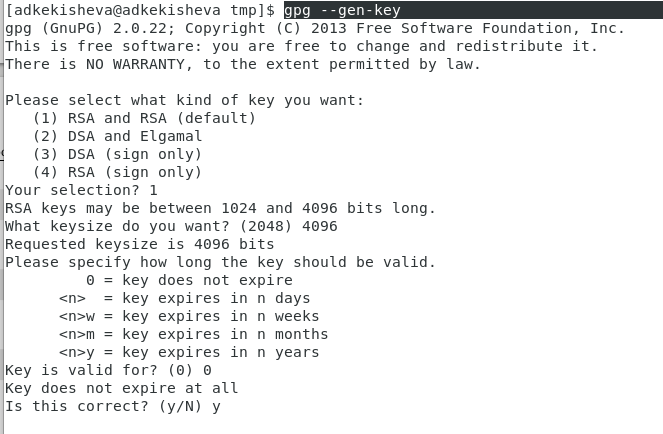
Базовая настройка git и команда создания клча SSH

1. Просмотрела созданный ключ SSH (рис. @fig:025).



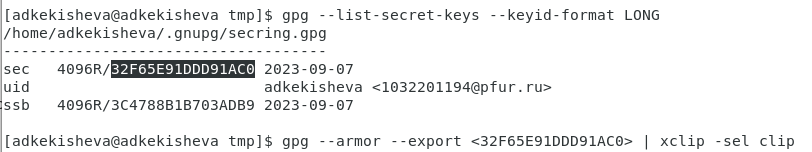
Ключь SSH

1. Сгенерировала ключ GPG командой gpg –ful-gen-key (рис. @fig:026).



Команда генерации RSA ключа

1. Вывела список ключей и скопировала отпечаток приватного ключа (рис. @fig:027)



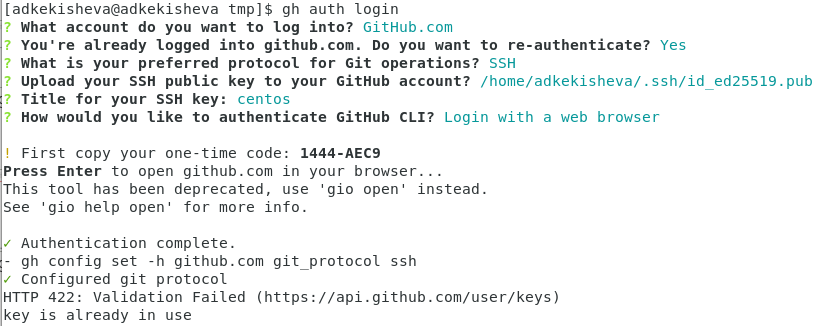
RSA ключ

1. Настроила автоматические подписи коммтитов (рис. @fig:028).

Настройка автоматических подписей коммтитов

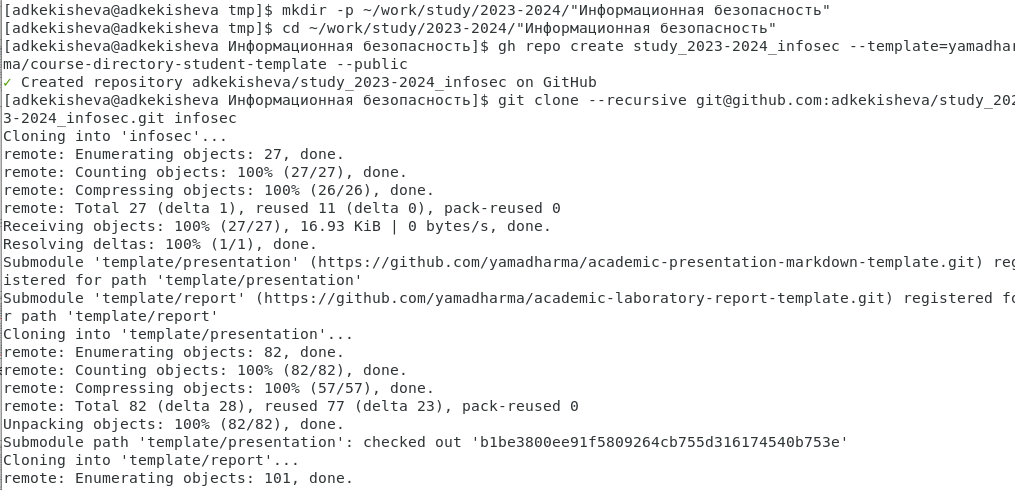
Настройка автоматических подписей коммтитов

1. Перешла к настройке gh и сначала авторизировалась, для этого ответила на несколько вопросов и подключилась в гит через браузер (рис. @fig:029).



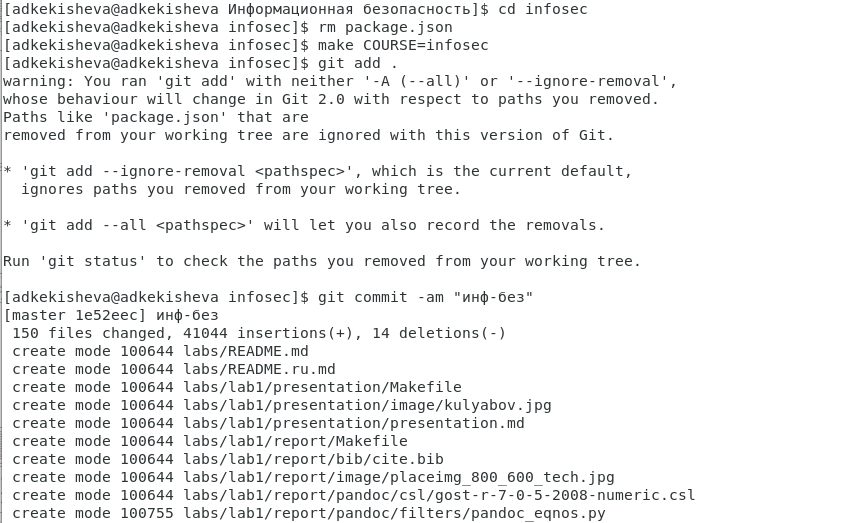
Авторицазия на github

1. Создала репозиторий курса на основе шаблона и вместе с тем локальный каталог для выполнения лабораторных работ (рис. @fig:030).



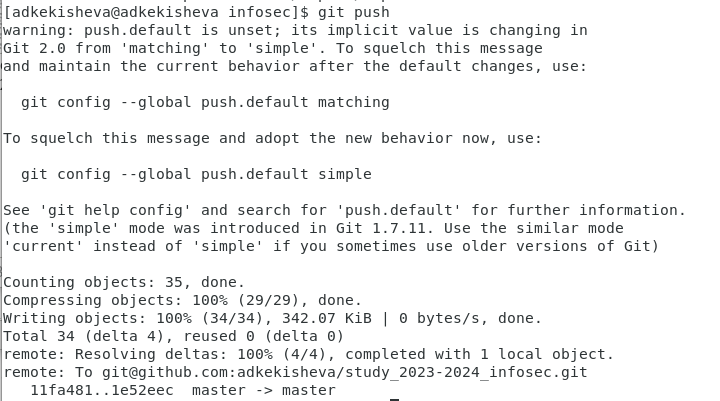
Создание репозитория курса на основе шаблона

1. Перешла в каталог курса infosec, удалила в нём лишшние файлы и создала необходимые каталоги (рис. @fig:031).



Удаление файлов json и создание папок для лабораторных

1. Отправила файлы на сервер (рис. @fig:032).



Отправка файлов в репозиторий

1. После выполнения отчёта преобразовала файлы из Markdown в pdf и docx.

# Выводы

* Приобрела практических навыки установки операционной системы на виртуальную машину, настройки минимально необходимых для дальнейшей работы сервисов.
* Поработала с git, вспомнила работу с ним, установила програмное обеспечение.

# Список литературы

1. [Пособие по установке виртуальной манины](https://esystem.rudn.ru/mod/page/view.php?id=1031497)
2. [Задания к лабораторным](https://esystem.rudn.ru/mod/folder/view.php?id=1031496)
3. [Администрирование систем Linux. Монтирование файловых систем](https://rus-linux.net/MyLDP/BOOKS/LSA/ch07.html)