

Лабораторная работа №1. Введение в Linux.

Что потребуется перед началом:

- ПК, способный запустить систему виртуализации с виртуальной машиной GNU/Linux.
- Минимум 6 GiB свободного места на жестком диске (под систему и снапшоты).
- Пакет или установщик системы виртуализации (рекомендуется VirtualBox).
- Загруженный образ дистрибутива (рекомендуется Ubuntu 20.04).

План и задачи лабораторной:

1. Часть 1. Начало работы
2. Подготовка рабочего окружения
3. Права, пользователи, su и sudo
4. Настройка сети ВМ
5. Подключение по ssh, ssh-agent
6. Установка и работа с tmux
7. Выполнение базовых команд
8. Часть 2. Продвинутая работа с системой
9. Автоматизируем сбор данных о системе с bash
10. Делаем сбор данных регулярным с cron
11. Пишем свой systemd-сервис
12. Запускаем процесс внутри нового пространства имен
13. Установка docker и запуск hello-world

Отчет - в любом читаемом формате (pdf, md, doc, docx, pages).

Обязательное содержимое отчета:

0. Фамилия и инициалы студента, номер группы, номер варианта
1. План и задачи лабораторной работы
2. Краткое описание хода выполнения работы
3. Приложить очищенный вывод `history` выполненных команд

Что нужно сделать, чтобы сдать лабораторную?

1. Выполнить все действия, представленные в методических указаниях и ознакомиться с материалом
2. Продемонстрировать результаты выполнения преподавателю, быть готовым повторить выполнение части задач из лабораторной по требованию

3. Ответить на контрольные вопросы

Вступление

Для полного понимания рекомендуется ознакомиться с разделами "Терминология", "Файлы, каталоги, права" и "Работа с системой" [справочных материалов](#).

Часть 1. Начало работы

Задача: установить любую систему виртуализации, настроить виртуальную машину с любым дистрибутивом GNU/Linux и выполнить набор простейших действий (подробно описан в п. 1.2).

Примечание: Если вы не работали с виртуализацией раньше - ставьте VirtualBox.

Вопросы различий виртуализации и решения проблем с системами виртуализации выходят за рамки курса.

Примечание: Если вы не работали раньше с GNU/Linux - ставьте дистрибутив Ubuntu (он используется в методических указаниях). Вопросы различий дистрибутивов и проблем с установкой ПО на разных дистрибутивах выходят за рамки курса.

Примечание: Лектор любит и всем советует использовать Debian, для более продвинутых есть опция работать по [материалам курса Linux](#).

Примечание: Если вы опытный "линуксоид" - стоит сразу начинать работать по ssh (сразу проделав)

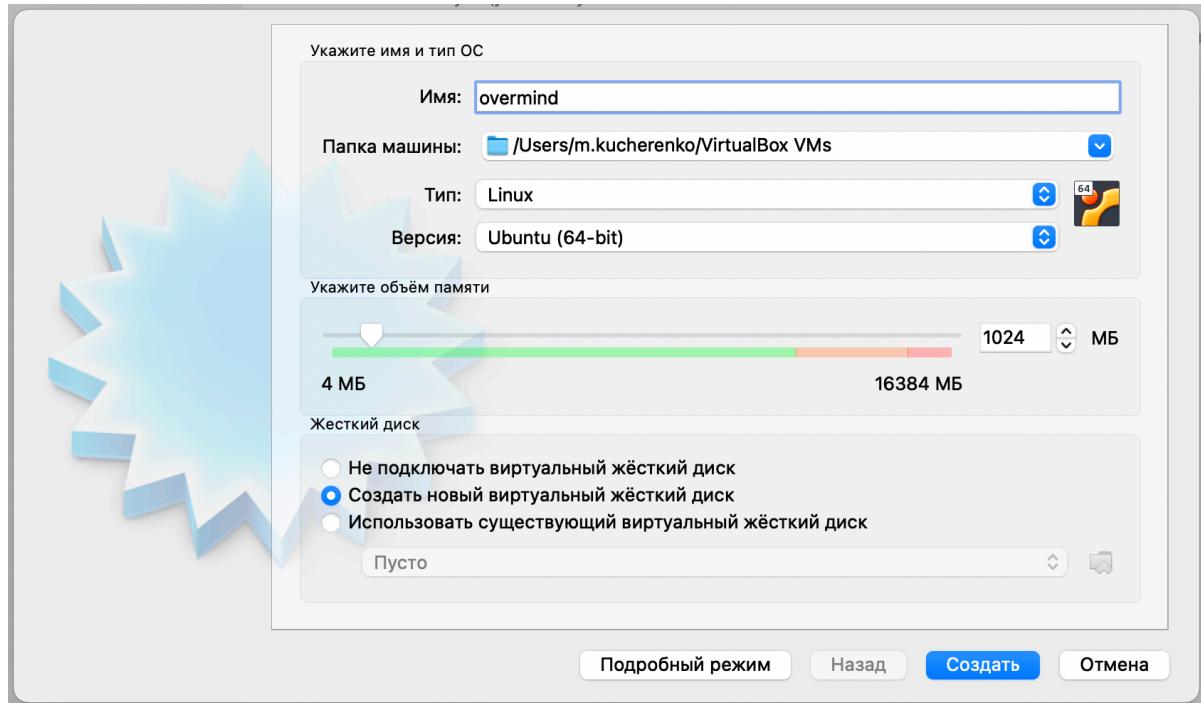
1.1. Подготовка рабочего окружения

0. Устанавливаем [VirtualBox](#) или любую другую систему виртуализации.
1. Скачиваем образ для установки дистрибутива Ubuntu 20.04 с [официального сайта](#).
Вы можете выбрать любой способ загрузки, подходящий вам, например через [Torrent-клиент](#).

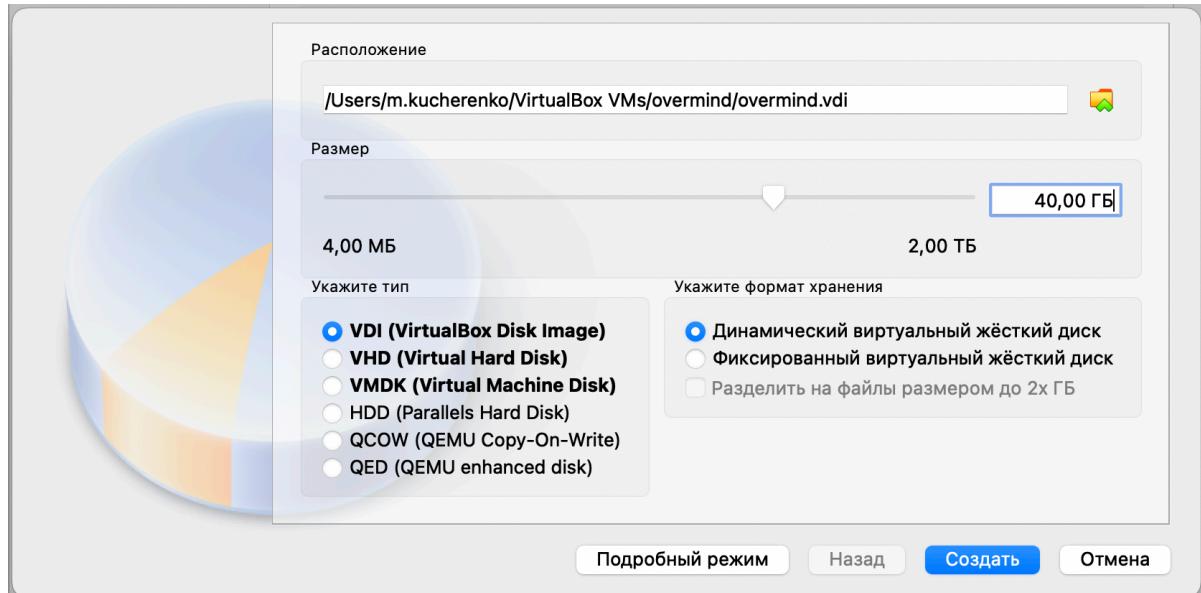
Во время подготовки курса использовался образ `ubuntu-20.04.5-live-server-amd64.iso`, как наиболее универсальный вариант. Но с серверов-зеркал он загружался целую вечность. **Лучше использовать BitTorrent**.

2. Пока качается образ - создаем виртуальную машину (далее VM) в системе виртуализации. Ставить будем в VirtualBox (далее VB).

Как назвать - дело ваше. Удобно набрать "Ubuntu" в названии, тогда VB сразу подставит значения для Ubuntu в выпадающие списки.



Выделяем диск. Использовать рекомендуется динамический виртуальный диск. Тогда покуда внутри ВМ вы не забьете место на диске - оно не будет занято и в вашей хостовой системе. Фактический размер будет динамически растягиваться под требования (но автоматически сжиматься не будет). Достаточно будет и 10 GiB. Лучше сразу больше, все равно место оно не занимает, пока не потребуется.



Получится как-то так:

The screenshot shows the Oracle VM VirtualBox Manager interface. At the top, there are tabs for 'Инструменты' (Tools), 'Создать' (Create), 'Настроить' (Configure), 'Сбросить' (Reset), and 'Показать' (Show). Below these are two main sections: 'Общие' (General) and 'Превью' (Preview).

Общие (General) settings for the VM 'overmind' (Ubuntu 64-bit):

- Имя: overmind
- ОС: Ubuntu (64-bit)
- Оперативная память: 1024 МБ
- Порядок загрузки: Гибкий диск, Оптический диск, Жёсткий диск
- Ускорение: VT-x/AMD-V, Nested Paging, Паравиртуализация KVM

Превью (Preview) shows a terminal window with the following text:

```
overmind@overmind:~$ cat /etc/issue
Ubuntu 12.04 LTS \n \l
overmind@overmind:~$
```

Below the General settings are other configuration sections:

- Дисплей** (Display):
 - Видеопамять: 16 МБ
 - Коэффициент масштабирования: 2.00
 - Графический контроллер: VMSVGA
 - Сервер удалённого дисплея: Выключен
 - Запись: Выключена
- Носители** (Devices):
 - Контроллер: IDE
 - Вторичное устройство IDE 0: [Оптический привод] Пусто
 - Контроллер: SATA
 - SATA порт 0: overmind.vdi (Обычный, 40,00 ГБ)
- Аудио** (Audio):
 - Аудиодрайвер: CoreAudio
 - Аудиоконтроллер: ICH AC97
- Сеть** (Network):
 - Адаптер 1: Intel PRO/1000 MT Desktop (Сеть NAT, 'linux-cluster')
- USB** (USB):
 - USB-контроллер: OHCI
 - Фильтры устройств: 0 (0 активно)
- Общие папки** (Shared Folders):
 - Отсутствуют
- Описание** (Description):
 - Отсутствует

3. Идем в настройки VirtualBox и создаем новую сеть NAT. Настраиваем проброс портов 127.0.0.1:2222 -> 10.0.2.15:22.

VirtualBox - Сеть

Общие Ввод Обновления Язык Дисплей Сеть Плагины Прокси

Сети NAT

Активна

Включить сеть

Имя сети: **linux-cluster**

CIDR сети: **10.0.2.0/24**

Опции сети:

Поддержка DHCP

Поддержка IPv6

Объявить маршрутом IPv6 по умолчанию

Проброс портов

Отмена OK

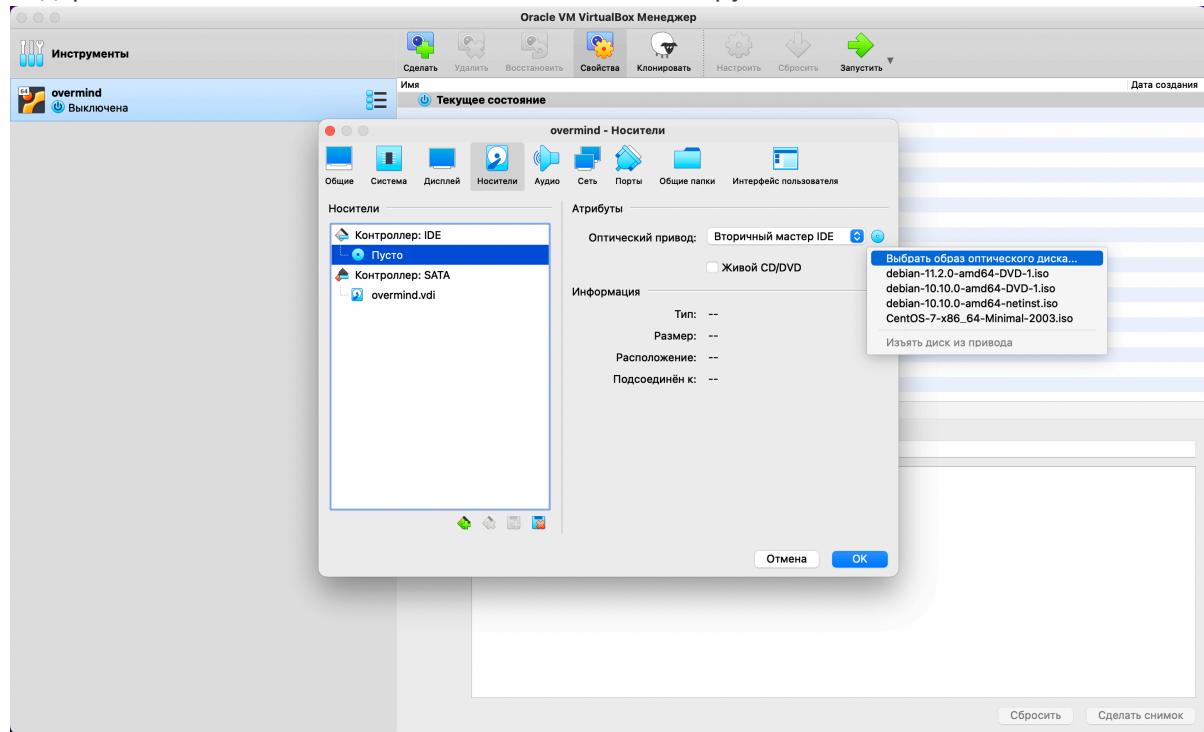
OK

IPv6

Протокол	Адрес хоста	Порт хоста	Адрес гостя	Порт гостя
TCP	127.0.0.1	2222	10.0.2.15	22

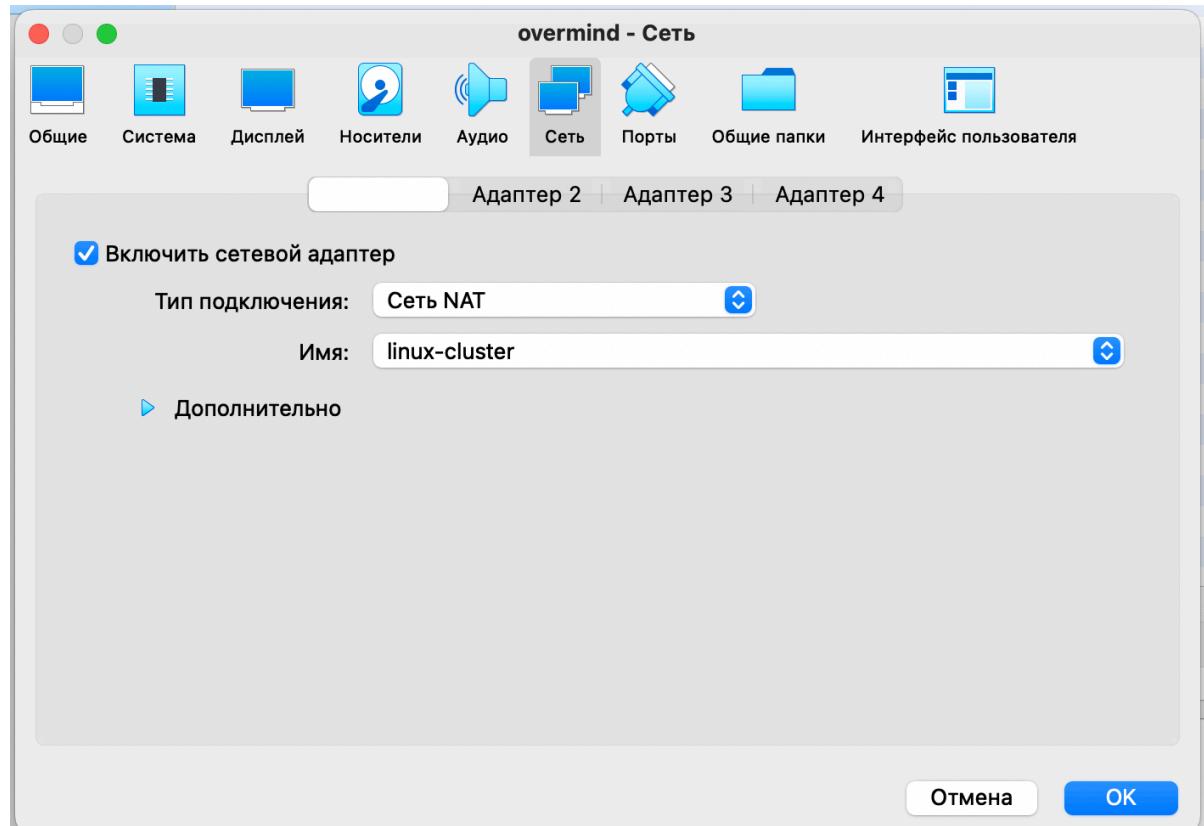
Отмена OK

- В настройках ВМ в системе виртуализации выбираем загруженный образ в качестве содержимого оптического носителя, чтобы с него загрузиться:



Примечание: Для систем с большим разрешением экрана советую сразу поставить в "Настройки ВМ" -> "Дисплей" коэффициент масштабирования побольше. На MacBook-ах с Retina-display иначе слишком мелко.

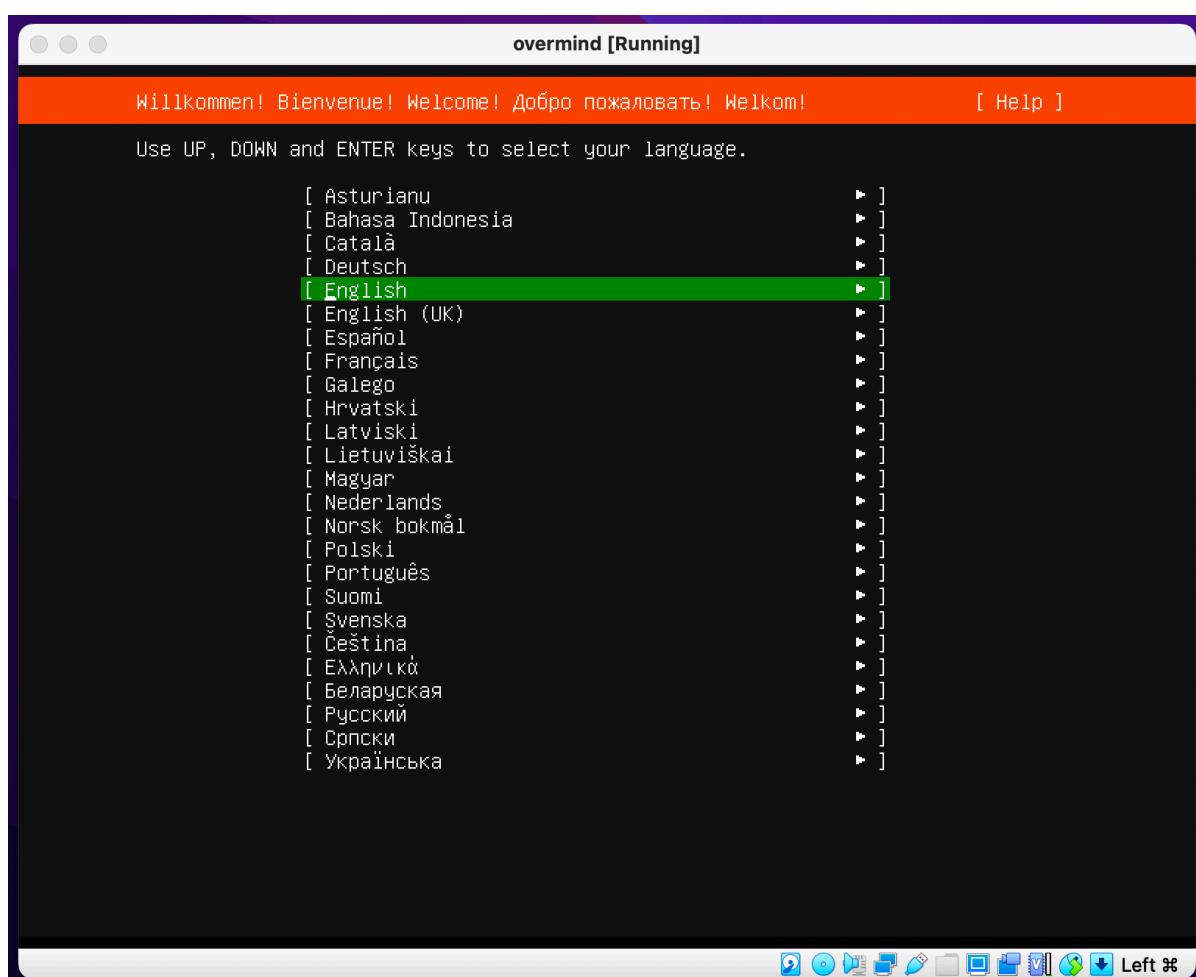
- В настройках ВМ в пункте "Сеть" выбираем "Сеть NAT" и нашу новую сеть.



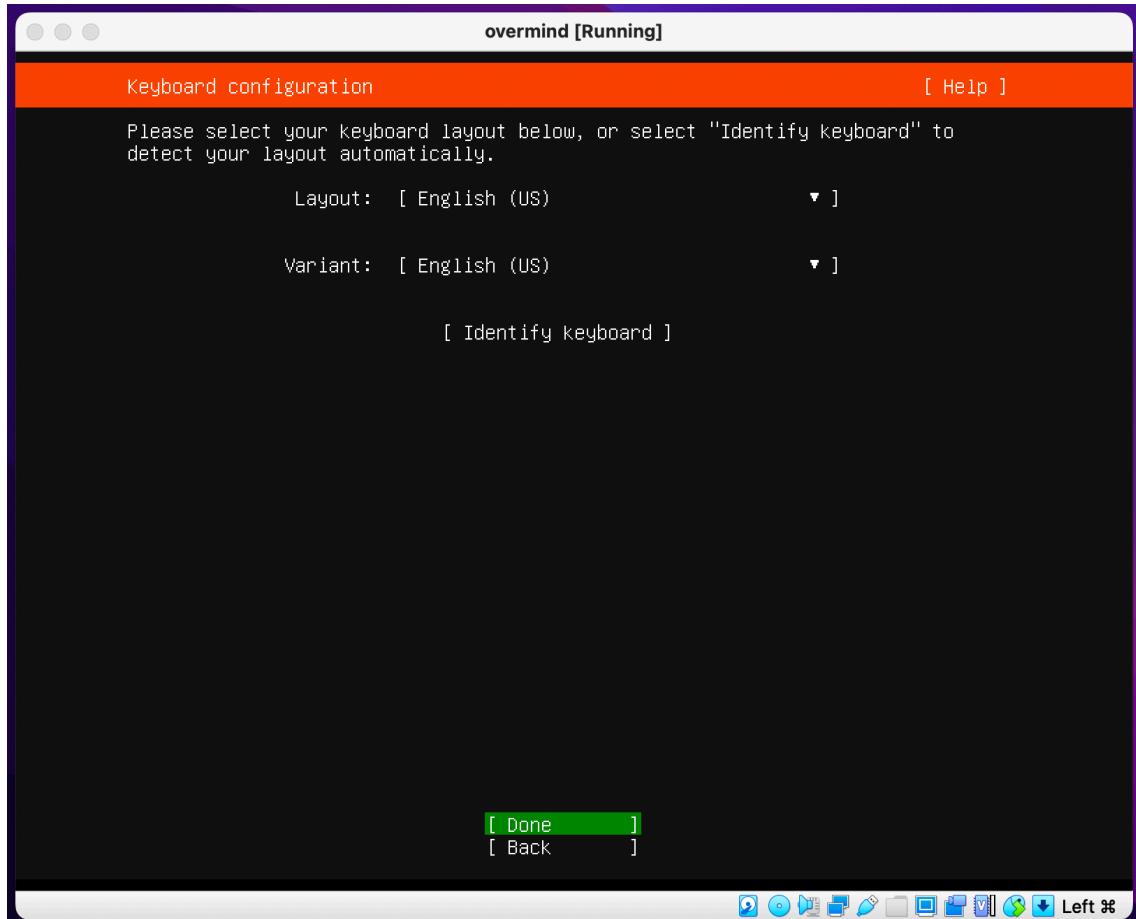
- Запускаем виртуальную машину и устанавливаем систему. Подробно рассматривать процесс установки не будем, остановимся только на самых важных деталях. [Руководство по установке Ubuntu-Server](#).

Для интересующихся руководство по установке Debian для x86-64 архитектуры вы можете найти [тут](#).

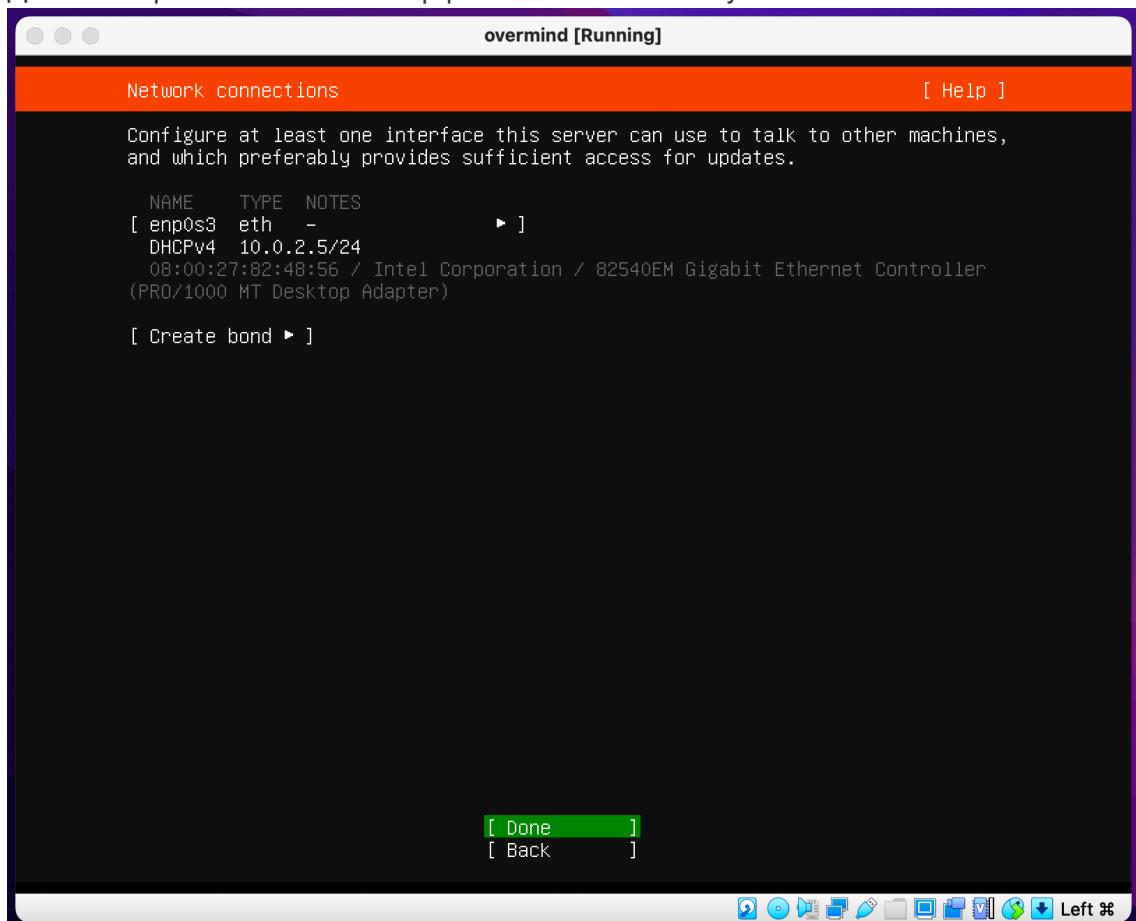
[Полный список руководств.](#)



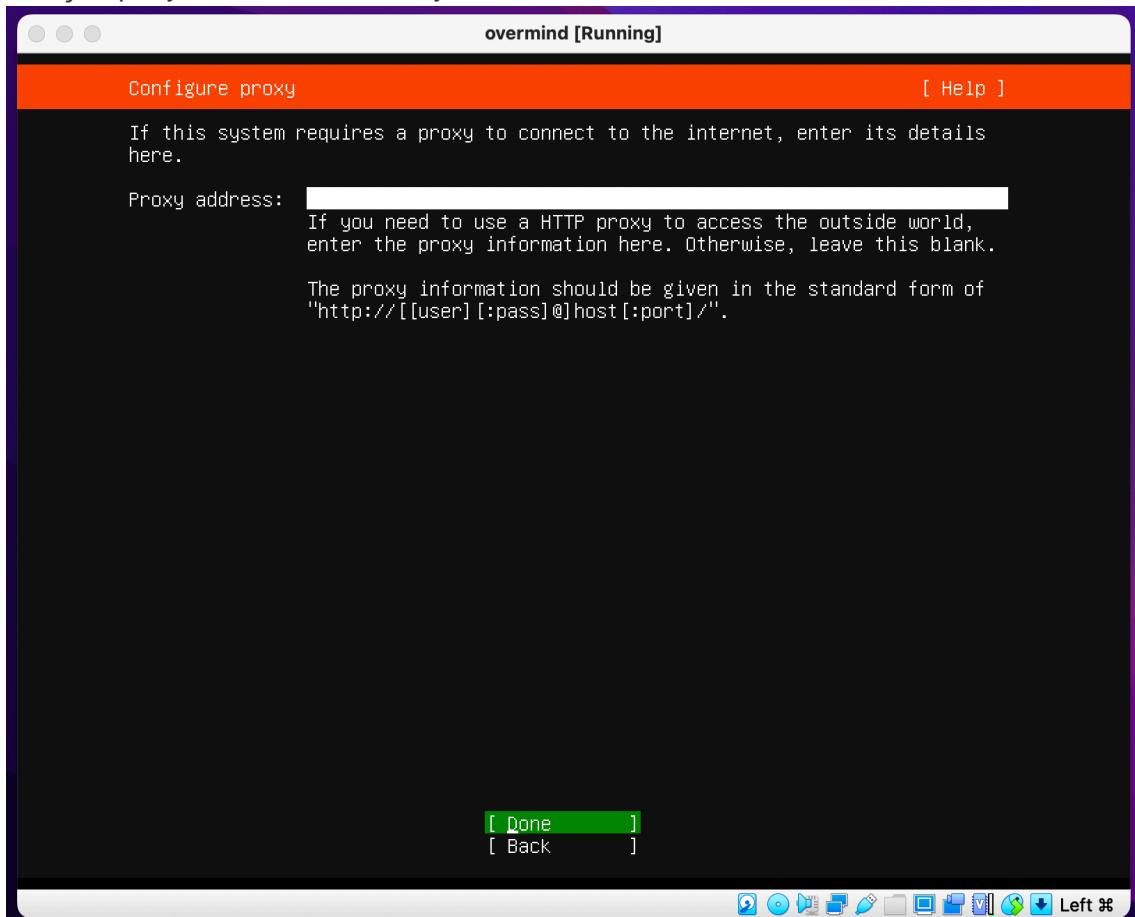
- Выбор региона и раскладки клавиатуры. Если выбираете себе что-то кроме английской раскладки - дело ваше, страдать Вам. Я оставляю все по-умолчанию.



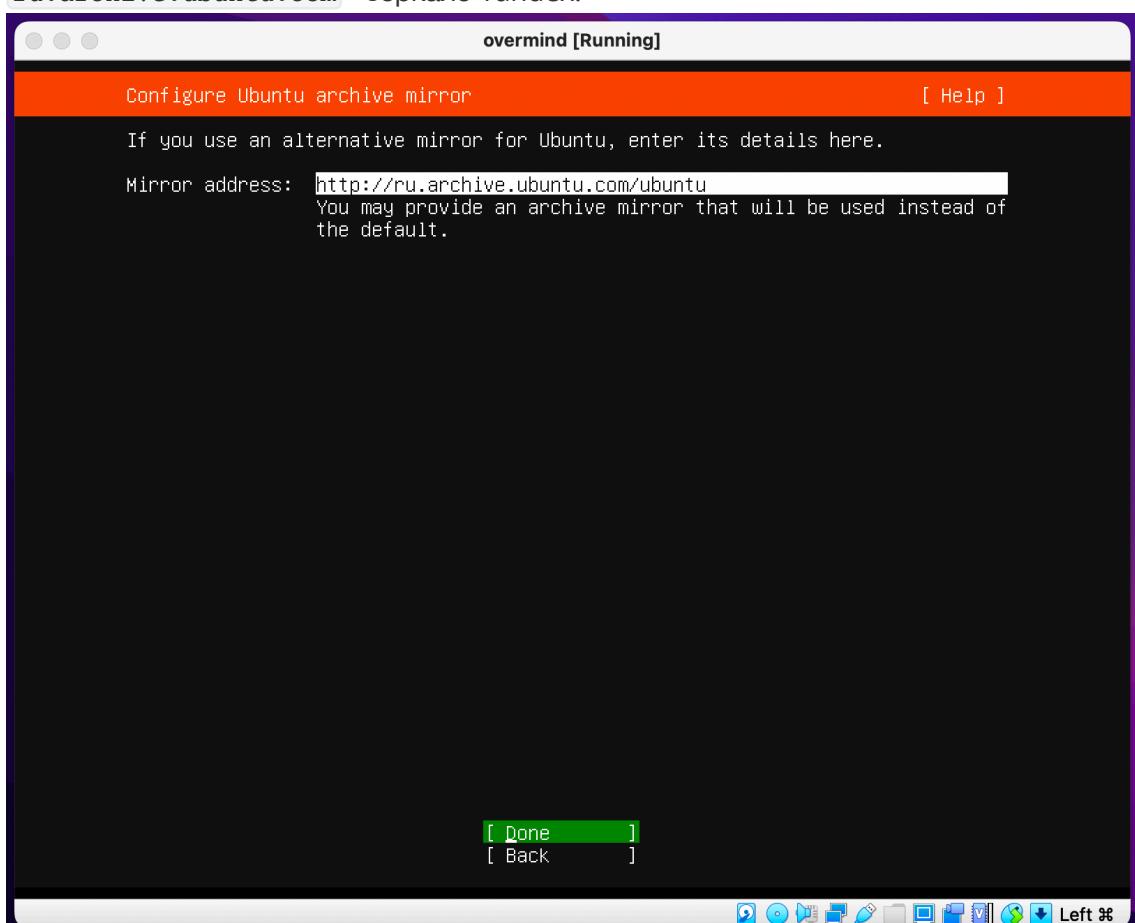
- Далее настройки сетевого интерфейса. Оставляем по-умолчанию.



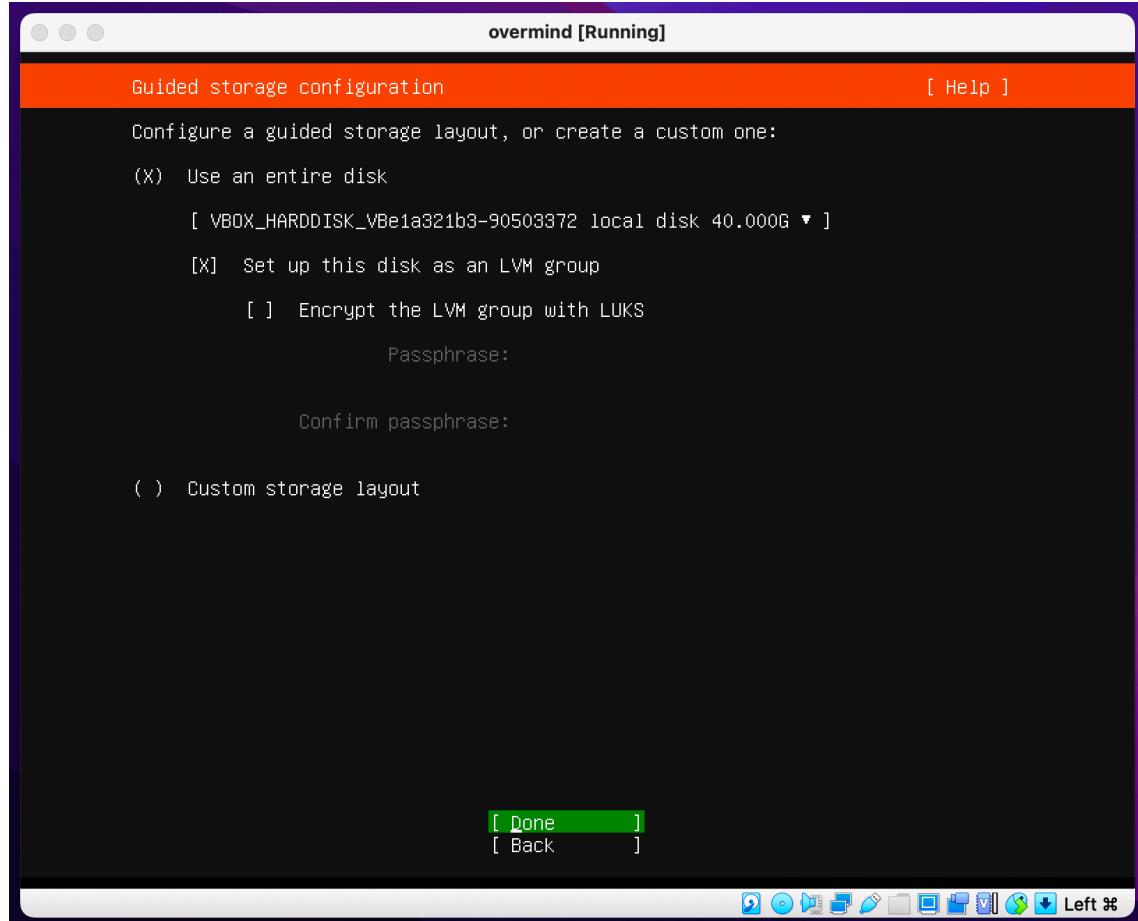
- Proxy - пропускаем, оставляем пустым.



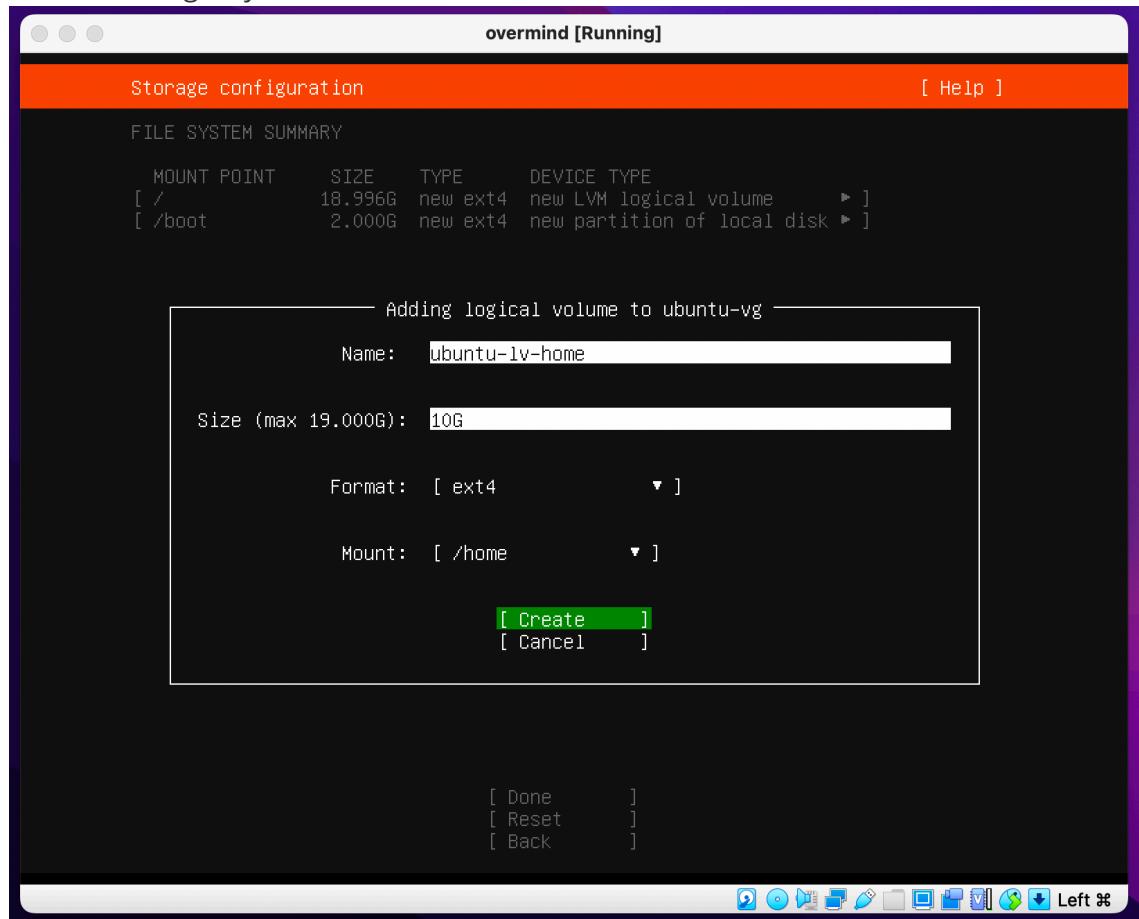
- Какое зеркало для загрузки репозиториев использовать - выбираем ru.archive.ubuntu.com - зеркало Yandex.



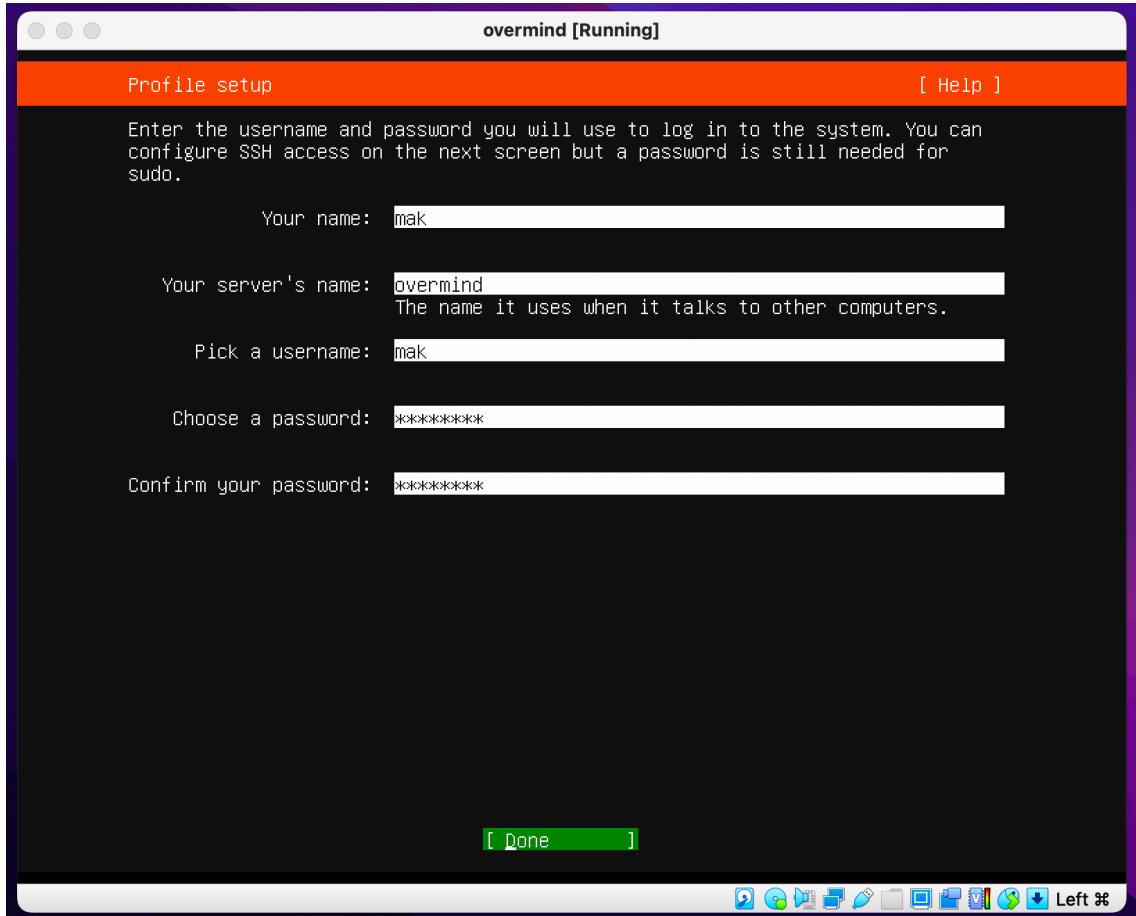
- Настройки разметки диска. Можем оставить по-умолчанию - система автомагически поставится поверх LVM. Для реальных систем идут жаркие споры о том стоит или не стоит ставить систему на LVM. Но у нас требований к надежности нашей виртуальной машины нет, поэтому установим сразу поверх. Из плюсов решения: можно легко расширять доступное под систему место с помощью добавления новых дисков в lvm. Из минусов: при потере части этих дисков есть шанс системе больше не запустить.



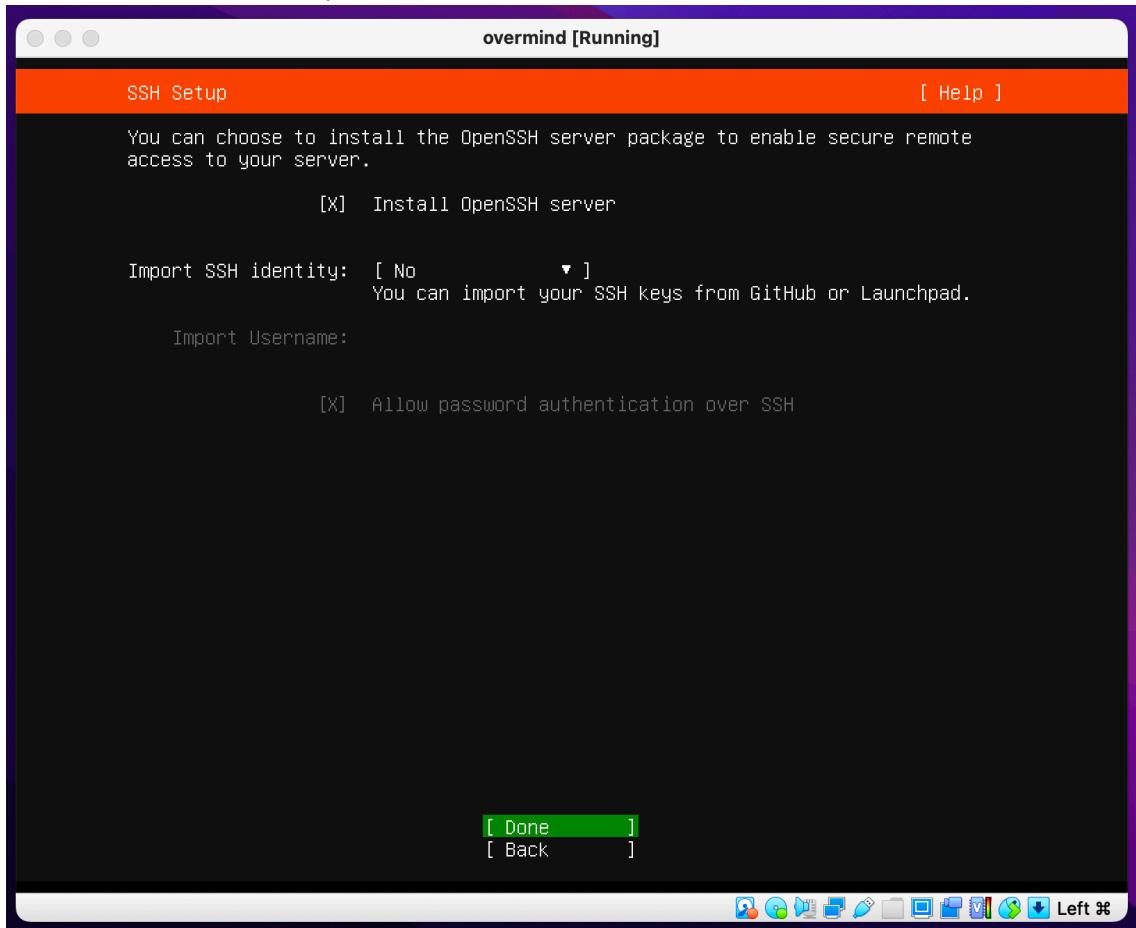
Но для желающих разобраться разберем добавление нового LV-раздела, выбрали Custom storage layout:



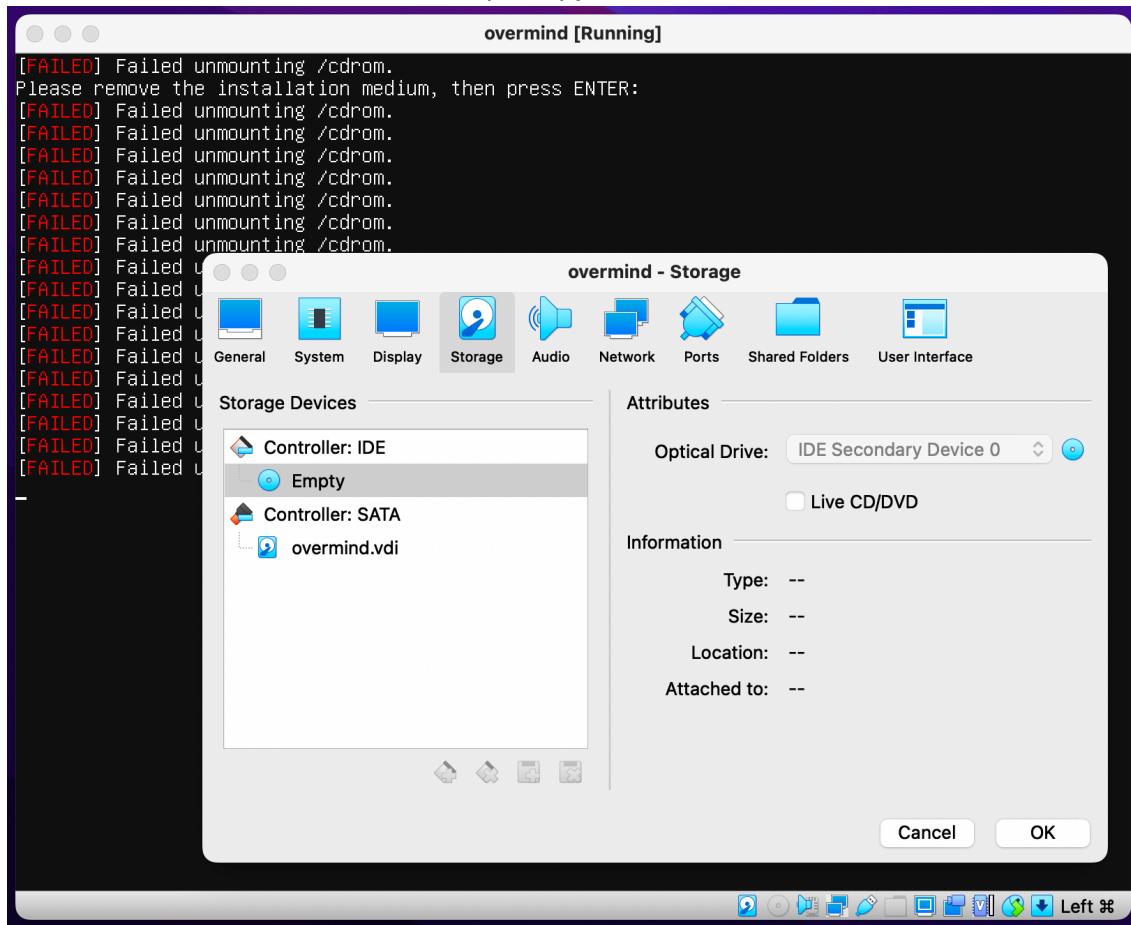
- Настройки пользователя, `hostname` - [сетевое имя системы](#) и пароля. Пароль [root](#) (пользователя с идентификатором (UID - user ID) 0 и имеющим неограниченные привилегии, аналог администратора, далее будем называть его root) не устанавливается. Задаем пароль только для своего основного пользователя. Настоятельно не советуем ставить свой настоящий сложный пароль на ВМ. После ввода его попросят еще раз для подтверждения, что вы в своем уме.



- Выбираем установить OpenSSH-сервер.



- Ожидаем завершения установки, перезагружаемся. После выбора пункта Reboot нужно будет зайти в настройки и проверить, что носитель из виртуального дисковода извлечен. После этого перезагружаем ВМ.



- ВМ "на логине". Пробуем войти своим пользователем. Если все введено верно, то вы увидите приглашение ввода команд (command prompt), например:

mak@overmind:~\$. Критический успех!

```

overmind (Netplan) [Running]
Ubuntu 20.04.5 LTS overmind tty1
overmind login: mak
Password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.4.0-125-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

 System information as of Thu 08 Sep 2022 08:53:35 AM UTC

 System load: 0.0 Processes: 106
 Usage of /home: 0.0% of 9.75GB Users logged in: 1
 Memory usage: 22% IPv4 address for enp0s3: 10.0.2.15
 Swap usage: 0%

1 update can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

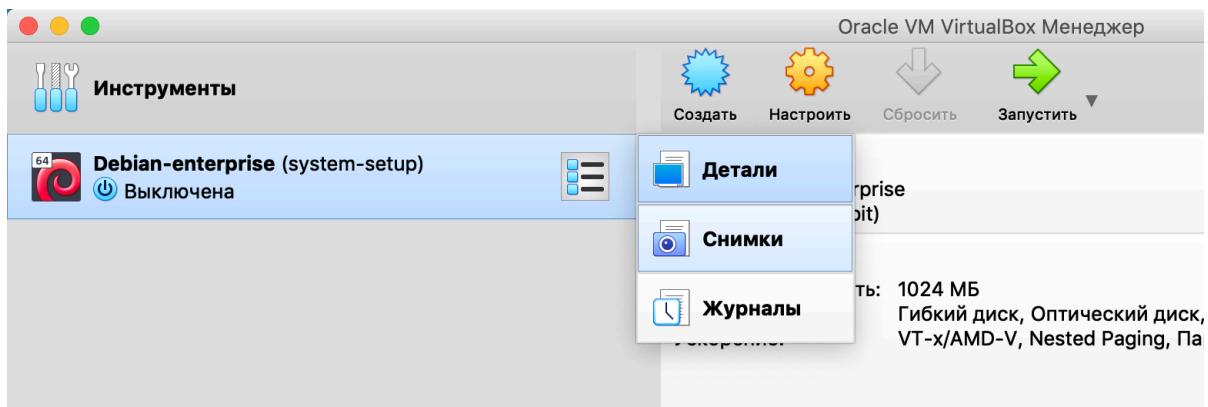
Last login: Thu Sep  8 01:47:19 UTC 2022 from 10.0.2.2 on pts/1
mak@overmind:~$ 

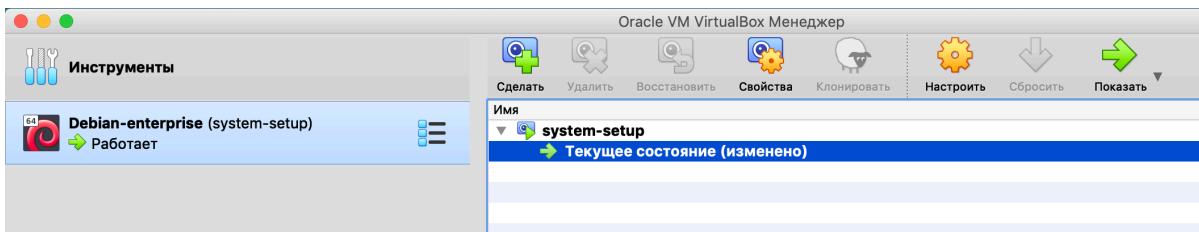
```

6. Делаем мгновенный снимок виртуальной машины, если ваша система виртуализации это поддерживает. Тогда в любой момент мы сможем вернуться на данное состояние системы, если что-то сломаем. В VB переходим во вкладку "Снимки" (Snapshots) и нажимаем "сделать". Названия не принципиальны. Для примера сделали это над другой ВМ (Debian).

Люди делятся на два типа:

- 1) Еще не делают бэкапы
- 2) Уже делают бэкапы

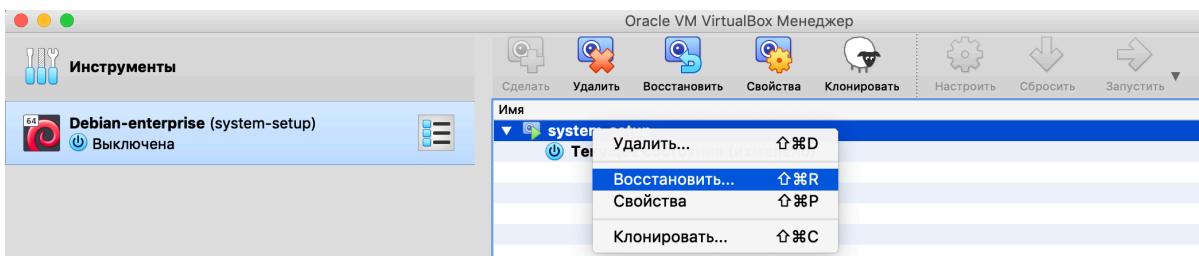




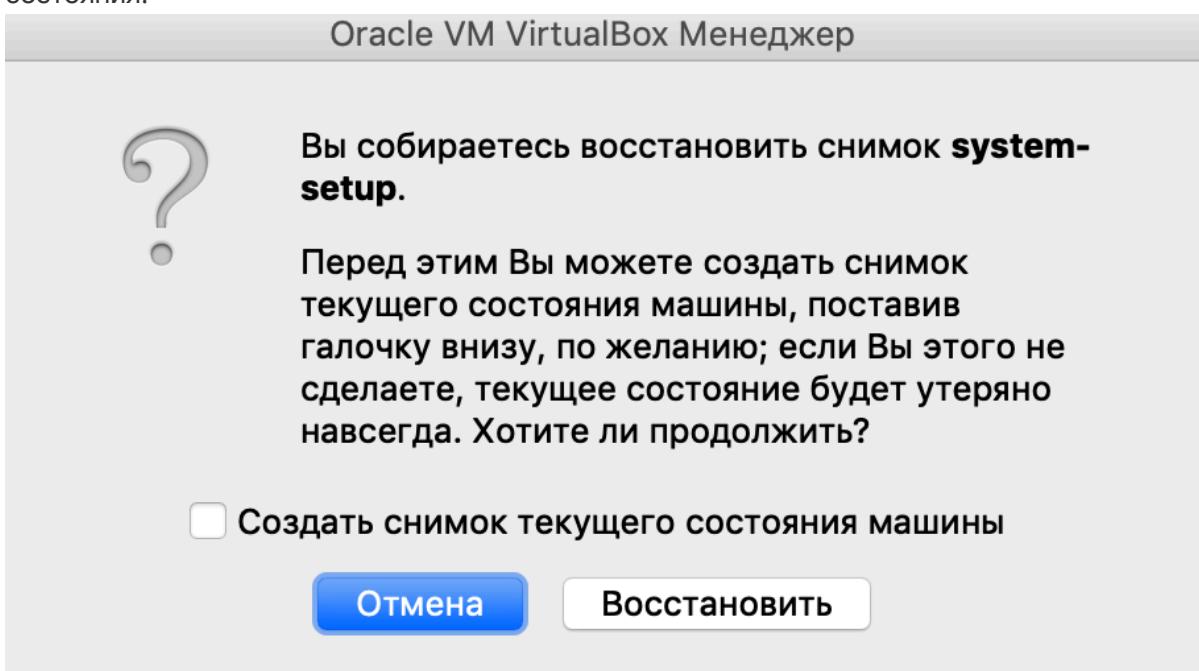
7. Что-нибудь изменим в системе, например напишем пару команд и выключим ВМ. Теперь попробуем восстановить мгновенный снимок из шага 5. Если все в полном порядке, вы откатитесь к состоянию ВМ на момент конца шага 4.

Люди, которые делают бэкапы, делятся на два типа:

- 1) Еще не проверяют свои бэкапы после создания
- 2) Уже проверяют свои бэкапы после создания

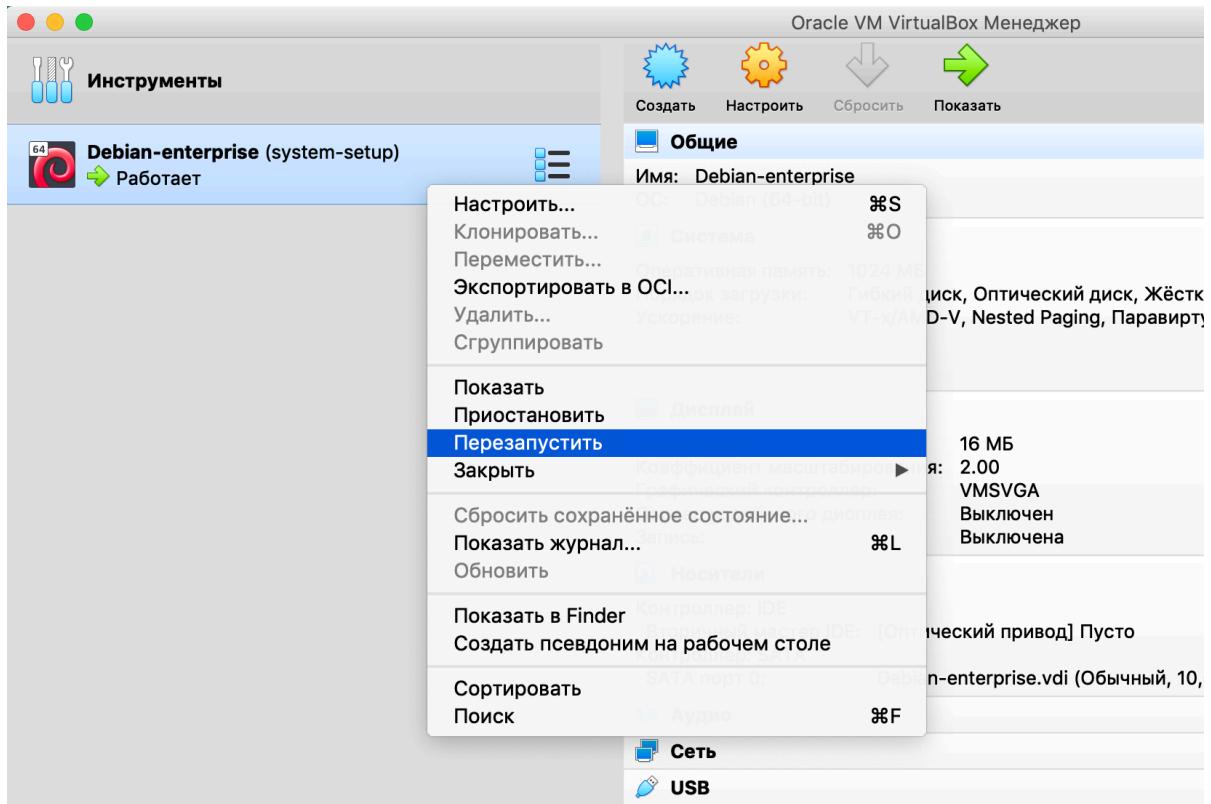


Снимаем галочку, чтобы не создать снимок текущего (будем считать его испорченным) состояния:



Можете добавить в список навыков в выводе отчета, что вы научились делать резервные копии и сразу проверять их на работоспособность. **Очень важный навык для любого DevOps-инженера или администратора.**

- Перезагружаемся (reset) с помощью VB. ВМ мгновенно перезапустится (будто мы обесточили и снова запустили сервер). Если все в порядке - вы снова окажетесь "на логине".



1.2. Права, пользователи, su и sudo

Становимся root.

Станьте суперпользователем (root-ом) с помощью утилиты `sudo -i`. Так как пароль root не задан - мы пока не можем стать им с помощью `su`.

Root (англ. *root* — корень; читается «рут»), или **суперпользователь** — это специальный аккаунт и группа пользователей в UNIX-подобных системах с идентификатором UID 0 (User IDentifier), владелец которого имеет право на выполнение всех без исключения операций. Суперпользователь UNIX-систем имеет логин «root» только по умолчанию и легко переименовывается при необходимости, часто встречается переименование в «toor» для усложнения подбора паролей автоматическими сканерами.

su (англ. *Substitute User, Set UID, Switch User, Super User* — замена пользователя, переключение пользователя, суперпользователь) — команда Unix-подобных операционных систем, позволяющая пользователю войти в систему под другим именем, не завершая текущий сеанс. Обычно используется для временного входа суперпользователем для выполнения административных работ. По умолчанию предполагается работа от имени пользователя root. Если первый аргумент `su` — дефис `-`, среда будет установлена такой же, как при регистрации заданного пользователя. Иначе передается текущая среда, за исключением значения \$PATH, которое задается переменными PATH и SUPATH в файле `/etc/default/su`.

sudo (англ. **S**ubstitute **U**sers and **d**o, дословно «подменить пользователя и выполнить») — программа для системного администрирования UNIX-систем, позволяющая делегировать те или иные привилегированные ресурсы пользователям с ведением протокола работы. Основная идея — дать пользователям как можно меньше прав, но при этом достаточных для решения поставленных задач.

Редактируем sudoers.

Далее с помощью утилиты `visudo` можно отредактировать файл `/etc/sudoers` (через утилиту - чтобы ничего не сломать, она создает временный файл и перед применением изменений проверяет синтаксис) - разрешить группе `wheel` делать `sudo` с паролем.

Примечание: в GNU/Linux CTRL часто обозначается как ^

Сохранить в nano: `^O`

Выйти из nano: `^X`

Поиск по файлу: `^W`

Этого можно добиться добавив строку `%wheel ALL=(ALL:ALL) ALL`

- **%wheel ALL=(ALL:ALL) ALL**

Первое поле показывает имя пользователя или группы, к которым будет применяться правило, в данном случае к группе **wheel**, для пользователя нужно убрать `%`.

- **%wheel ALL=(ALL:ALL) ALL**

Первое “ALL” означает, что данное правило применяется ко всем хостам.

- **%wheel ALL=(ALL:ALL) ALL**

Данное “ALL” означает, что пользователь группы **wheel** может запускать команды от лица всех пользователей.

- **%wheel ALL=(ALL:ALL) ALL**

Данное “ALL” означает, что пользователь группы **wheel** может запускать команды от лица всех групп.

- **%wheel ALL=(ALL:ALL) ALL**

Последнее “ALL” означает, что данные правила применяются всем командам.

Хорошая статья о sudoers файле [тут](#).

Добавим пользователя, группу и изменим права

Пользователи в Linux.

1. Создаем пользователя `ansible` с домашней директорией `/home/ansible/`. Например, с помощью утилиты `adduser`. Хорошая [статья](#) о создании пользователей.
2. Меняем пользователю `ansible` пароль с помощью `passwd ansible`.
3. Меняем `root` пароль с помощью `passwd`.
4. Создаем группу `wheel` с помощью `groupadd wheel`.

Важный момент: в GNU/Linux нет магии, все происходит максимально логично. Так, добавление пользователя сводится к изменению файлов `/etc/passwd` и `/etc/shadow`.

Исторически файл `/etc/passwd` содержал информацию о пользователях и их зашифрованные (чаще всего - хешированные) пароли. Однако, для того, чтобы пользователь мог взаимодействовать с системой - запускать программы от лица других

пользователей, просматривать права доступа и так далее, нам необходимо, чтобы этот файл был доступен для чтения всем. А значит, в теории, пароли других пользователей можно было бы подобрать перебором, ведь они нам были бы тоже доступны. Для того, чтобы избежать проблем с утечкой паролей - их вынесли в отдельный файл `/etc/shadow`. Больше о формате этих файлов стоит узнать в справочных материалах.

Формат записей файла /etc/passwd:

Формат записей файла /etc/shadow:

```
mak:$6$TelW:18727:0:99999:7:::  
[-] [-----] [---] | [---] ||||  
| | | | +-----> 9. Неиспользованное поле  
| | | +-----> 8. Срок годности  
| | +-----> 7. Период бездействия  
| +-----> 6. Период предупреждения  
| +-----> 5. Максимальный возраст пароля  
| +-----> 4. Минимальный возраст пароля  
| +-----> 3. Последнее изменение пароля  
+-----> 2. Зашифрованный пароль  
+-----> 1. Имя пользователя
```

4. Создать директорию `/admin` с помощью `mkdir /admin`.
 5. [Читаем про систему прав](#)
 6. Сделать owner-ом директории `/admin` пользователя `ansible` и выдать права группе `wheel`:
`chown ansible:wheel /admin`
 7. Выдать на права на чтение и редактирование пользователю `ansible`, а группе `wheel` только не чтение:
`chmod 640 /admin`

History

Настроим для root сохранение истории, добавим это в конец `/root/.bashrc`, чтобы история незамедлительно сливалась в файл (а не при завершении сессии):

```
shopt -s histappend  
PROMPT_COMMAND="history -a; $PROMPT_COMMAND"
```

Аналогичное действие проведите для `.bashrc` вашего пользователя, с помощью которого вы подключаетесь на хост.

Для синхронизации history между сессиями можно добавить в текущий env (либо также добавить в `.bashrc`, но не советую, т.к. при вводе нескольких команд с разных консолей одновременно - может перетирать историю, которая не успела сохраниться на диск, но если вас не очень заботит мультипоточность можно):

```
export PROMPT_COMMAND='history -a;history -c;history -r'
```

А теперь перестаньте уже быть `root`, это вредно - наберите команду `exit` или нажмите комбинацию `^D` (CTRL+D)

Демонстрация некоторых команд на некотором сервере, просто для ознакомления, не руководство к действию:

```
[root@enterprise:~# adduser ansible
Adding user `ansible' ...
Adding new group `ansible' (1001) ...
Adding new user `ansible' (1001) with group `ansible' ...
Creating home directory `/home/ansible' ...
Copying files from `/etc/skel' ...

[New password:
[Retype new password:
passwd: password updated successfully
Changing the user information for ansible
Enter the new value, or press ENTER for the default
[     Full Name []: AnsibleName
[     Room Number []: NoRoom
[     Work Phone []: NoPhone
[     Home Phone []: NoPhone
[     Other []:

[Is the information correct? [Y/n] y
[root@enterprise:~#
[root@enterprise:~# groups ansible
ansible : ansible
[root@enterprise:~# usermod -aG wheel ansible
usermod: group 'wheel' does not exist
[root@enterprise:~#
[root@enterprise:~# addgroup wheel
Adding group `wheel' (GID 1002) ...
Done.
[root@enterprise:~# usermod -aG wheel ansible
[root@enterprise:~#
[root@enterprise:~# groups ansible
ansible : ansible wheel
[root@enterprise:~#
[root@enterprise:~# addgroup admin
Adding group `admin' (GID 1003) ...
Done.
[root@enterprise:~# mkdir /admin
[root@enterprise:~# chown ansible:admin /admin
[root@enterprise:~# ls -la /admin
total 8
drwxr-xr-x  2 ansible admin 4096 Jan 29 22:26 .
```

1.3. Настройка сети ВМ

Ubuntu за последние годы сменила уже 3 или 4 сервиса управления сетью, не советую привязываться к текущему - netplan.

Смотрим на свои сетевые интерфейсы и списки маршрутов:

```
ip a # сокращение от ip address  
ip r # сокращение от ip route  
  
# Для любителей старины, стоит знать:  
apt install net-tools # ставим пакет старых утилит управления сетью  
ifconfig # аналог ip a  
route -n # аналог ip r
```

Запомните свой IP и GW:

```
$ ip a | grep -v '127.0.0.1' | grep 'inet '  
    inet 10.0.2.15/24 brd 10.0.2.255 scope global noprefixroute enp0s3  
$ ip r  
default via 10.0.2.1 dev enp0s3 src 10.0.2.15 metric 202
```

IP - 10.0.2.15

GW - via 10.0.2.1

Редактируем настройки:

```
sudo nano /etc/netplan/00-installer-config.yaml
```

Содержимое файла должно быть примерно таким (детали зависят от настроек сетевых интерфейсов выше):

```
network:  
  ethernets:  
    enp0s3:  
      dhcp4: no  
      addresses: [10.0.2.15/24]  
      gateway4: 10.0.2.1  
      nameservers:  
        addresses: [8.8.8.8,8.8.4.4]  
  version: 2
```

Применяем настройки:

```
sudo netplan --debug apply
```

1.4. Подключение по ssh, ssh-agent

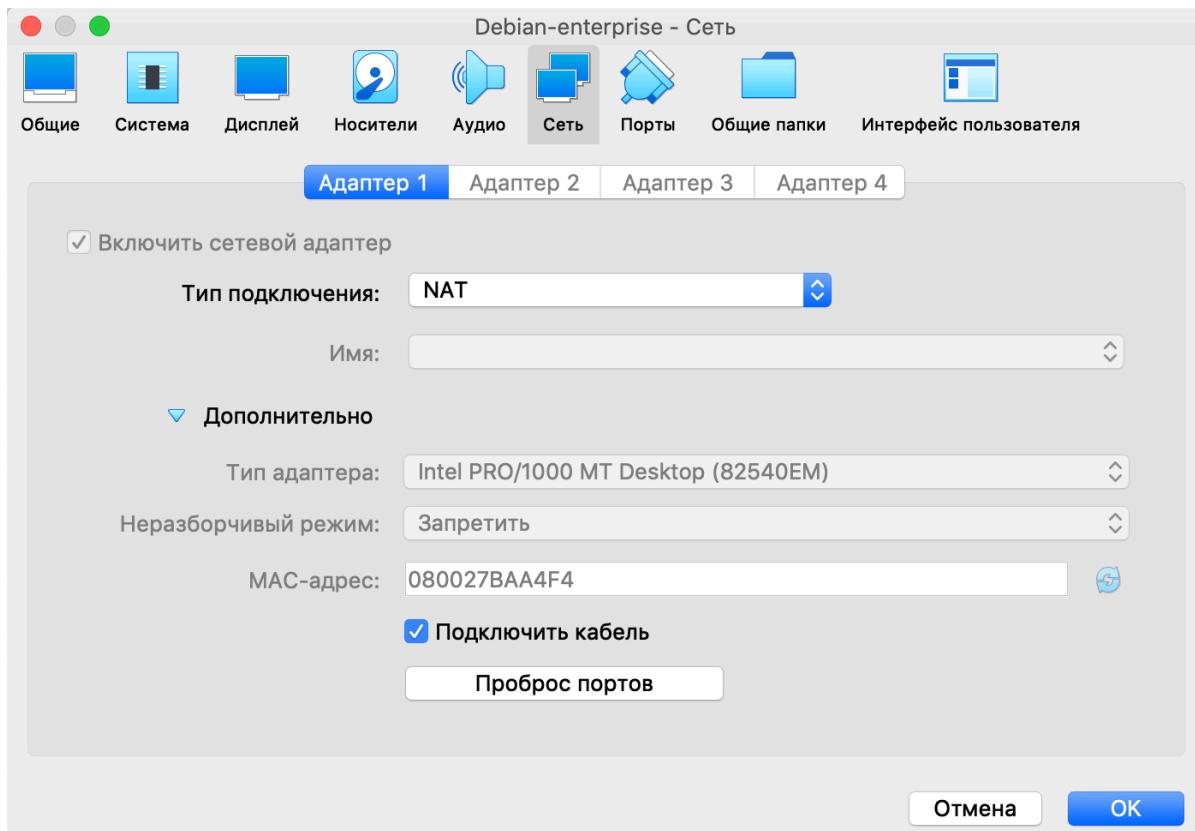
SSH (англ. **S**ecure **S**hell — «безопасная оболочка») — сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и [туннелирование](#) TCP-соединений (например, для передачи файлов). Схож по функциональности с более старыми протоколами Telnet и rlogin, но, в отличие от них, шифрует весь трафик, включая и

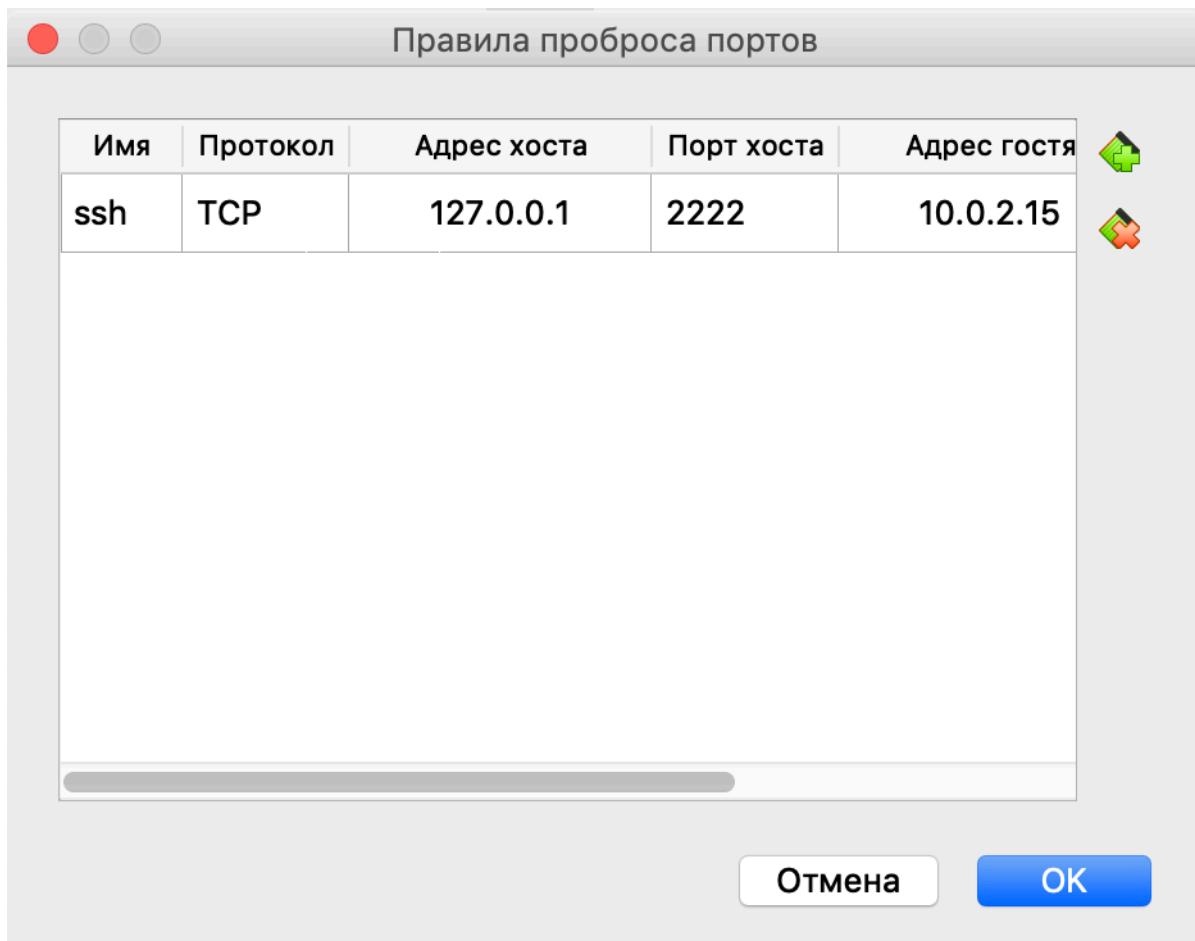
передаваемые пароли. SSH допускает выбор различных алгоритмов шифрования. [Больше про SSH](#).

По-умолчанию ssh-сервер использует 22-ой TCP-порт.

0. Установить на **основную систему** (ее еще называют хостовой или гипервизором) ssh-клиент. В Unix/Linux системах ssh стоит по-умолчанию. В Windows - [PuTTY](#), проще всего скачать установщик всех утилит (сам PuTTY, PuTTYgen, pageant и прочие программы).
1. **Мы уже сделали это в пункте 1.1.** Но покажу еще один вариант как настроить "проброс портов" (port-forwarding) для виртуальной машины **127.0.0.1:2222 -> 10.0.2.15:22**.

Зачем это нужно? Мы используем NAT, соответственно нам нужно направить пакеты за NAT в нашу виртуальную сеть. Рассматривайте это как конфигурирование настроек роутера, для доступа к вам в локальную сеть из интернета.



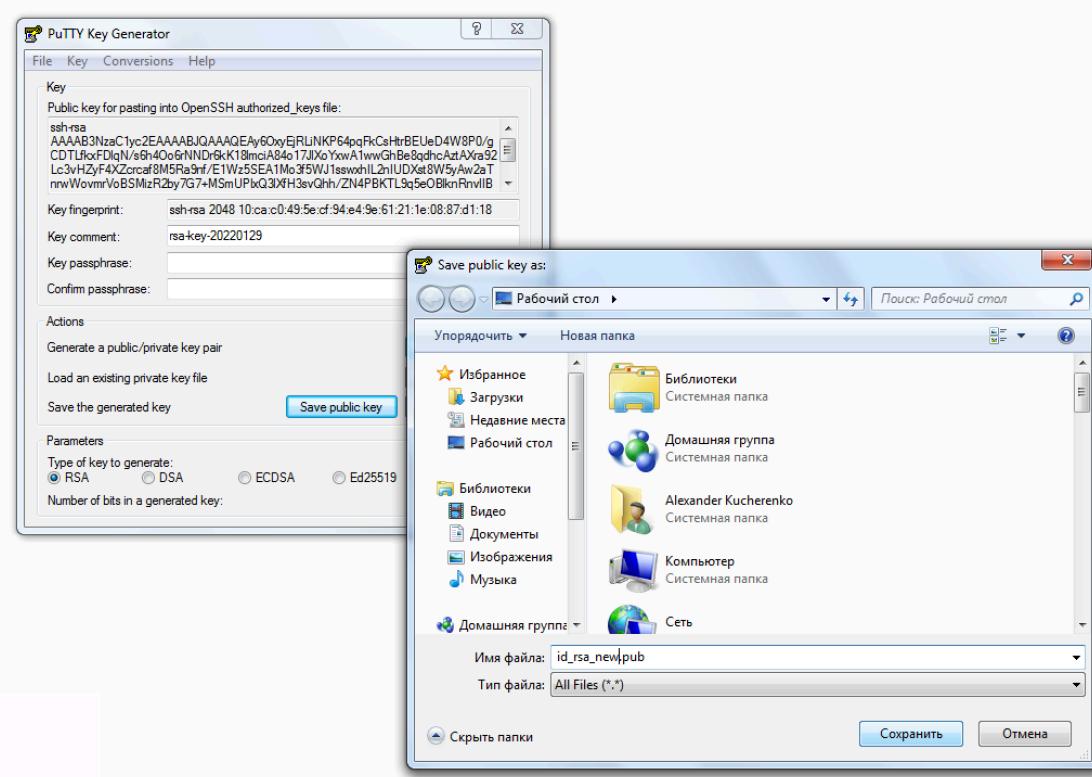
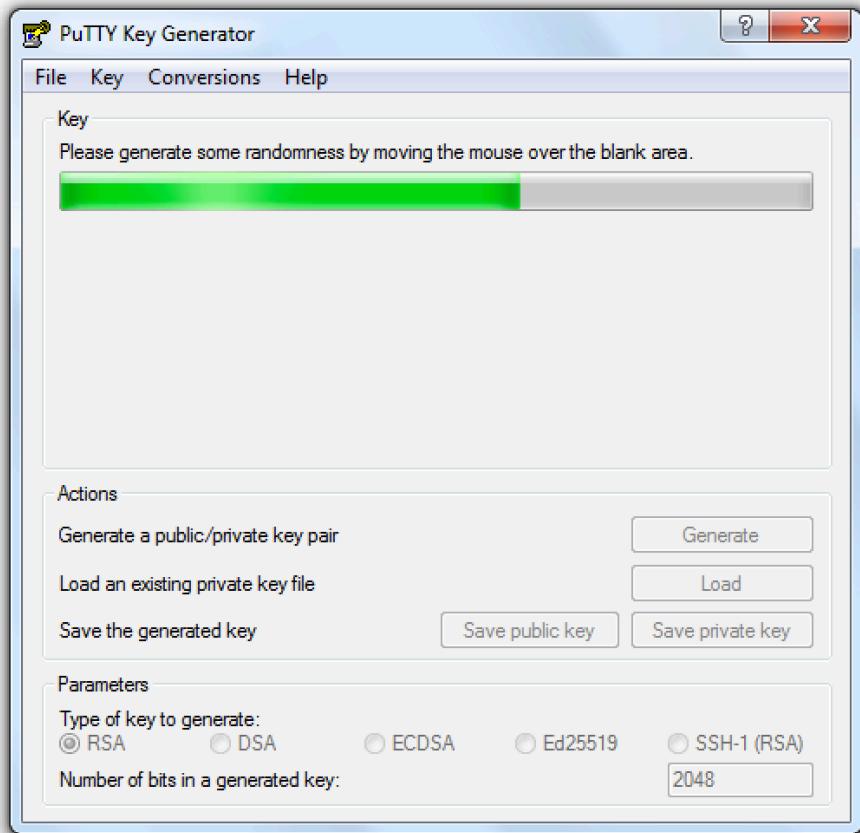


2. На **ВМ** - установить и запустить sshd (проверить запущен ли: `systemctl status sshd`, запустить: `systemctl start sshd`)
3. На **основной** системе настроить ssh-подключение по ключу

Примечание: подробная статья о PuTTY есть на [хабре](#).

- Сгенерировать ssh-ключи (в утилите [PuTTYgen](#) или [ssh-keygen](#)). Для этого нужно перегенерировать ключ 2 версии.

Windows:



Unix/Linux:

```

(base) mbp-mak-home:~ snipghost$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/snipghost/.ssh/id_rsa): id_rsa_new
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_rsa_new.
Your public key has been saved in id_rsa_new.pub.
The key fingerprint is:
SHA256:HVw/YlJXIU/Y0Xlc5j+F0K3oNtZuqboal/XHI2RdkA snipghost@mbp-mak-home
The key's randomart image is:
+---[RSA 2048]---+
|       .ooE+=I
|       . o.o.B.I
|       o oo%+=I
|       . .o=oX*I
|      S ... .+=I
|       o B.oI
|       o = ool
|       . o ... .I
|       ..++ .o.   I
+---[SHA256]---+
(base) mbp-mak-home:~ snipghost$ ll id_rsa_new*
-rw----- 1 snipghost  staff  1831 29 янв 23:31 id_rsa_new
-rw-r--r-- 1 snipghost  staff    404 29 янв 23:31 id_rsa_new.pub

```

- Положить публичную часть ключа в файл `/home/ansible/.ssh/authorized_keys` на **ВМ**

Для простоты можно скопировать публичную часть ключа из окна генератора в notepad, а потом сразу забросить на сервер руками - подключитесь с помощью PuTTY, откройте текстовый редактор на редактирование нужного файла и скопируйте из notepad в окно терминала ключ.

Да, директории `.ssh` может еще не существовать, создайте. Права установите в `rwx-----`.

ВНИМАНИЕ! Вам нужна публичная часть ключа в формате однострочной записи:

`<type> <key> <optional-comment>`.

Пример:

```

ssh-rsa
XXXXXXXXXXc2EAAAADAXABAAVAXC5Am7+fGZ+5zXBGgXS6GUvmsXCLGc7tX7/rViXk3
+eShZzaXnt75gUmT1I2f75zFn2h1AIDGKwf4g12KWcZxy81TniUOTjUsVlwPymXUXxES
L/UfJKfbdstBhTOdy5EG9ryWA0K43SJmwPhH28BpoLfXXXXXG+/ilsXXXXXgRLiJ2W1
9MzXHp8z3Lxw7r9wx3HaV1P4XiFv9U4hGcp8RMI1MP1nNesFlOBpG4pV2bJRBTXNxeY4
16F8WZ3C4kuf8XxOo08mXaTpVZ3T1841altnNTZCcpKXuMrBjYSJbA8proXAXNwiivyo
e3X2KMXXXXdXXXXXXXXXXXX/ user@myserver

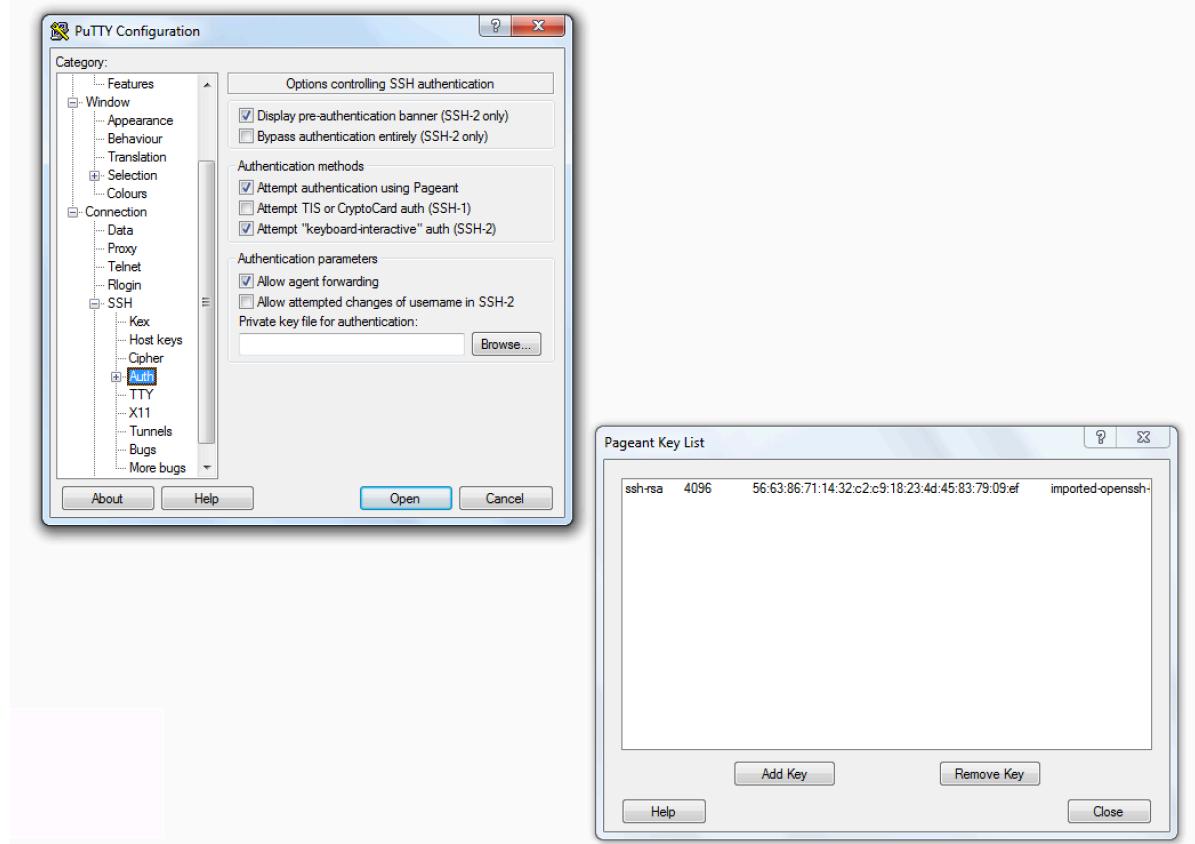
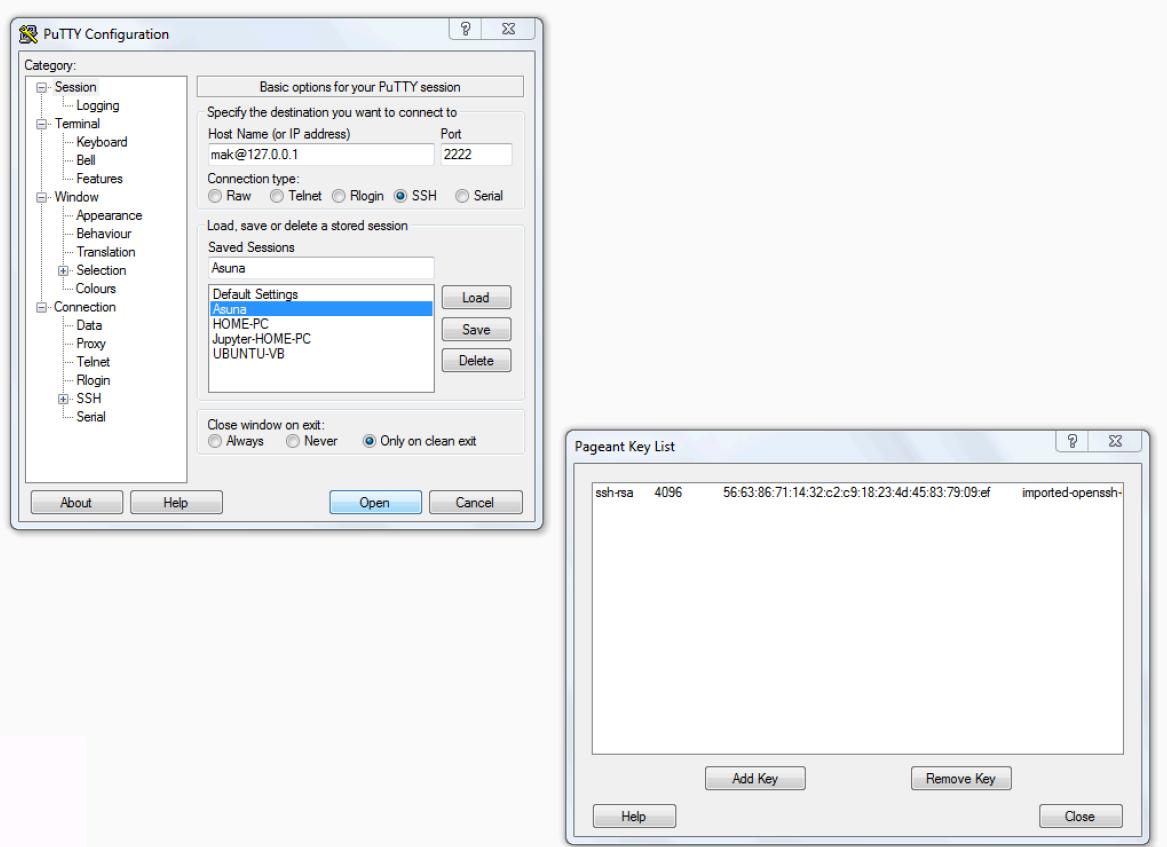
```

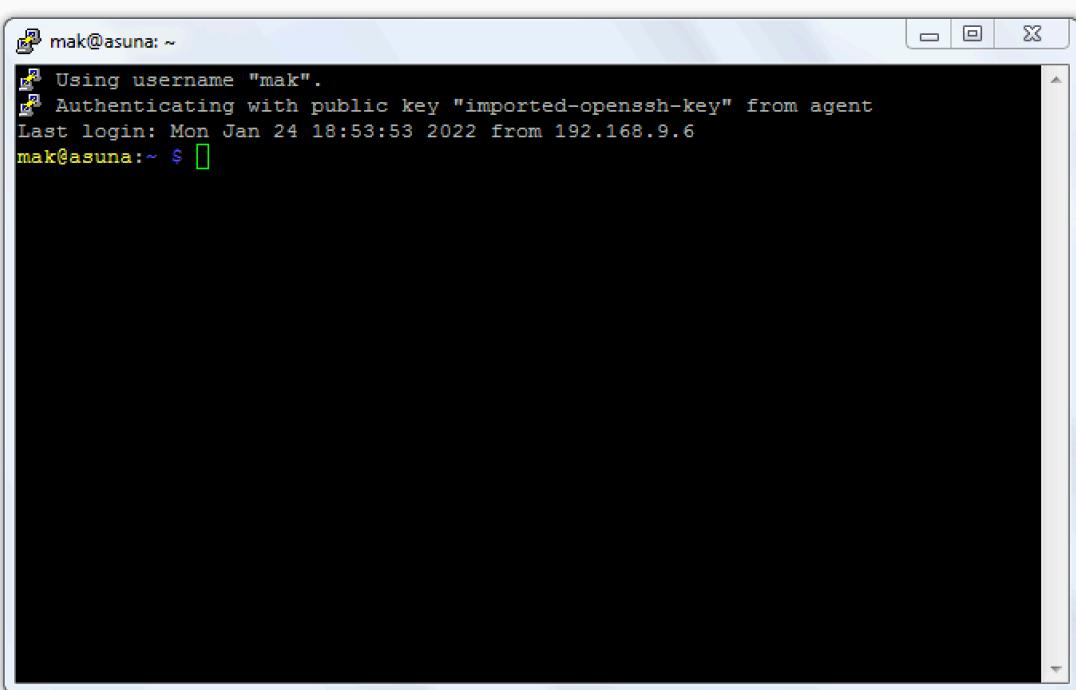
PuTTY сохраняет `.pub/. pem` файл в формате PKCS8. Для конвертации на Linux можно воспользоваться утилитой `ssh-keygen -i -m PKCS8 -f public-key`

- Проверить права файла `/home/ansible/.ssh/authorized_keys`, они должны быть `rw-r--r--`

- Продемонстрировать подключение по ssh (PuTTY+pageant для windows) под пользователем `ansible` и стать root-ом с помощью sudo. Подключаться на проброс портов: `<your_user>@127.0.0.1:2222`.

Windows:





Unix/Linux, сразу с применением `ssh-copy-id` вместо ручного редактирования

`authorized_keys`:

```
(base) mbp-mak-home:~ snipghost$ ssh-copy-id mak@127.0.0.1 -p 2222
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/Users/snipghost/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
mak@127.0.0.1's password:

Number of key(s) added:      1

Now try logging into the machine, with:  "ssh -p '2222' 'mak@127.0.0.1'"
and check to make sure that only the key(s) you wanted were added.

(base) mbp-mak-home:~ snipghost$ ssh -p 2222 mak@127.0.0.1
Linux enterprise 5.10.0-11-amd64 #1 SMP Debian 5.10.92-1 (2022-01-18) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*-/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Jan 29 19:03:23 2022
mak@enterprise:~ $
```

1.5. Установка и работа с tmux

Мы уже сталкивались с установкой пакетов. Рассмотрим подробнее, в Ubuntu для этого используется пакетный менеджер apt. Подробнее [тут](#).

```
sudo apt install tmux
```

Tmux - терминальный мультиплексор. Очень удобная штука для работы с несколькими сессиями в системе (или несколькими системами). Позволяет открывать и поддерживать на сервере несколько сессий. Что гарантирует защиту от внезапного завершения процессов в вашей консоли при разрыве соединения без использования `nohup`. Статья [тут](#).

```
(0) - cluster: 10 windows (attached)
(1) -> + 0: asuna*2 (2 panes)
(2) -> 1: asuna (1 panes) "root@asuna: ~"
(3) -> 2: asuna-mak- (1 panes) "asuna"
(4) -> 3: miku (1 panes) "root@miku: ~"
(5) -> 4: itsuki (1 panes) "root@itsuki: ~"
(6) -> 6: rikka (1 panes) "root@rikka: /usr/local/bin/format_tester"
(7) -> 7: metroid (1 panes) "root@metroid: /usr/local/bin/lcdapi"
(8) -> 11: ichika (1 panes) "root@ichika: ~"
(9) -> 12: ichika (1 panes) "root@ichika: ~"
(M-a) -> 13: nino (1 panes) "root@nino: ~/EmailParser"

0 (sort: index)
top - 01:07:37 up 97 days, 18:45, 22 users, load average: 0,38, 0,46, 0,42
Tasks: 212 total, 1 running, 211 sleeping, 0 stopped, 0 zombie
%Cpu(s): 5,8 us, 3,2 sy, 0,0 ni, 90,1 id, 0,4 wa, 0,0 hi, 0,5 si, 0,0 st
MiB Mem : 7875,3 total, 2899,6 free, 2417,7 used, 2558,0 buff/cache
MiB Swap: 100,0 total, 100,0 free, 0,0 used. 4879,3 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
17419 ceph 20 0 477284 62424 19984 S 3,3 0,8 959:48.15 ceph-mgr
380 message+ 20 0 6796 3748 3084 S 2,3 0,0 696:48.39 dbus-daemon
627 consul 20 0 1058028 238972 69672 S 2,3 3,0 4863:11 consul
628 consul 20 0 885320 22568 8952 S 1,7 0,3 242:39.58 consul_exporter
1 root 20 0 35240 8684 6 1,3 0,1 377:37.98 systemd
686 ceph 20 0 935892 650144 28 0 1,0 8,1 1195:36 ceph-mon
14186 root 20 0 10432 2892 2 1,0 0,0 0:01.67 top
1033 root 20 0 984720 24028 18756 S 0,7 0,3 525:21.96 docker-containe
10108 mak 20 0 12484 7156 2440 S 0,7 0,1 4:24.58 tmux: server
11 root 20 0 0 0 0 S 0,3 0,0 117:52.69 ksoftirqd/0
12 root 20 0 0 0 0 I 0,3 0,0 239:44.92 rcu_sched
340 root 20 0 0 0 0 S 0,3 0,0 109:15.38 jbd2/sda2-8
342 root 20 0 0 0 0 S 0,3 0,0 48:50.37 jbd2/sda3-8
621 root 20 0 1015340 61360 28232 S 0,3 0,8 593:50.09 dockerd
671 haproxy 20 0 19660 9788 2068 S 0,3 0,1 659:06.41 haproxy
722 root 20 0 25284 2692 1776 S 0,3 0,0 93:01.64 keepalived

GNU nano 3.2
/etc/prometheus

# Ansible managed
#
global:
  scrape_interval: 15s # Set the scrape interval to every X seconds. Default is every 1 minute
  evaluation_interval: 15s # Evaluate rules every X seconds. The default is every 1 minute
# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Victoria Metrics itself.
scrape_configs:
  - job_name: 'victoria-metrics'
    static_configs:
      - targets: ['localhost:8428']

  - job_name: 'node'
    static_configs:
      - targets:
          - 'nino:9100'

# Помощь   Записать   Поиск   Вырезать   Выровнять
```

[cluster] 0:asuna*2 1:asuna 2:asuna-mak- 3:miku 4:itsuki 6:rikka 7:metroid 11:ichika 12:ichika 13:nino "root@asuna: ~" 01:07 09-фев-22

Теперь запускаем сессию с идентификатором по-умолчанию (для первой - 0):

tmux

Создадим пару новых окон несколько раз нажав: $\text{Ctrl} + \text{B} + \text{C}$.

Переключимся между ними с помощью $\text{Ctrl} + \text{W}$. Переключаться между двумя последними можно $\text{Ctrl} + \text{L}$, переключиться на конкретное окно по его номеру, например первое, так: $\text{Ctrl} + \text{B} + \text{0}$.

Далее работать будем так. Если нужно отключиться без завершения сессий набираем $\text{Ctrl} + \text{D}$.

Закрыть окно можно с помощью $\text{Ctrl} + \text{C}$ - так мы перейдем в командный режим (ввод команд tmux-у в поле ввода внизу), затем вводим `kill-window` и нажимаем enter.

Для повторного подключения к нужной сессии `tmux a -t $ID`:

tmux a -t 0

1.6. Выполнение базовых команд

Шпаргалка наиболее частых команд:

- cd (change directory) - сменить рабочий каталог
- mv (move) - переместить объект фс
- cp (copy) - скопировать объект фс
- mkdir (make directory) - создать каталог
- echo - вывод аргументов на экран

- `cat` - вывод объекта (например, файла) на экран
- `less` - утилита для удобного просмотра больших текстовых файлов
- `sort` - утилита для сортировки
- `grep` - фильтрация строк по подстрокам

Статья о циклах в командной оболочке [тут](#)

Разберем несколько:

`ls` — покажет содержимое директории, в которой вы находитесь. Если после команды ввести адрес конкретной папки, то она покажет то, что хранится в ней. [Типы файлов. Система прав.](#)

Полезные аргументы:

- R** в выводе команды появятся файлы из поддиректорий
- I** отобразить списком с метаданными (права доступа и прочая информация из inode)
- a** отобразить скрытые (начинающиеся с точки) объекты

`cd` — change directory, сменить директорию. Из названия понятно, что с помощью этой команды можно перемещаться между каталогами. Вписываем её, а потом путь, например: `cd /path/to/dir`.

Есть сокращенные варианты:

- `cd -` для перемещения в предыдущую директорию
- `cd` (без аргументов) для прыжка в домашнюю папку

`cp` - сору - скопировать объект. Из полезных опций **-r** позволяет делать рекурсивно (для всех подкаталогов) и **-p** позволяет сохранить все права на файл как у исходного.

`pwd` — укажет полный путь до директории, в которой вы находитесь. Если вы перепрыгнули на рабочий стол с помощью `cd ~/Desktop/`, то `pwd` покажет что-то вроде `/home/mak/Desktop`.

`rm` — команда для удаления файлов, каталогов и их содержимого. Удаляет жесткие ссылки, фактически данные остаются на диске, просто мы теряем на них указатели и эти блоки данных теперь считаются пустыми. На форумах часто ходит злая шутка для новичков, которым советуют починить что-нибудь в их систему с помощью `rm -rf /`, что потенциально стирает все данные.

Полезные аргументы:

- r** удалить каталог и все вложенные объекты
- f** удалить без подтверждения

`grep` — это полнотекстовый поиск в файле. Очень полезная команда, когда вам нужно что-то найти. Например, `grep значение /путь/до/файла` покажет строки в которых есть «значение». А команда `grep -ril значение /путь/до/каталога/*` покажет все файлы, в которых есть искомое значение.

Полезные аргументы:

- r** рекурсивный поиск
- i** игнорировать регистр (case insensitive)
- I** вывести результат списком
- v** инвертировать работу - пропускать только строки, в которых **НЕТ** значения
- E** использовать регулярные выражения в "значении"

df — disk free - показывает данные по ФС, в т.ч. сколько еще свободно места на каких ФС.

-h human-friendly, переводит байты в читаемое

-i inode, показывает информацию по айнодам на ФС

du — disk usage - показывает данные по всем неудаленным файлам на ФС.

-h human-friendly, переводит байты в читаемое

chown пользователь:группа путь — команда позволяет изменить владельца файла или

каталога. **-R** для рекурсивного.

chmod права путь — изменяет права на файл или каталог. Подробнее в разделе "Система прав в GNU/Linux".

find — с помощью этой утилиты можно искать (и даже удалять) файлы по определённым параметрам. Например: `find /mnt/data/backup/ -mtime +3 -type f -iname "*sql.gz"` - выведет все файлы (`-type f`) в директории `/mnt/data/backup` 3-х дневной давности (дата изменения modify time +3 - `-mtime +3`) с именем попадающим в маску `*sql.gz`. При добавлении опции `-delete` удаляет найденные файлы. Часто используется в таком виде в скриптах для очистки старых логов или резервных копий.

awk — си-подобный сценарный язык построчного разбора и обработки входного потока по заданным шаблонам. Ключей много. Примеры и [статья на хабре](#):

`echo word1 word2 word3 word4 | awk '{ print $2,$4 }'` выведет `word2 word4` для каждой строки (тут всего одна).

`echo word1:word2:word3:word4 | awk -F: '{ print $1,$3 }'` выведет `word1 word3`.

tail — позволяет вывести только конец файла/потока.

+2 - выкинуть первую строку

-n X - оставить только X последних строк

head — аналогично tail, только для начала файла/потока.

ps — посмотреть список процессов, с аргументом `aux` показывает все процессы в системе.

Pipeline.

Для выполнения работы вам непременно потребуется знание о том, что такое [конвейер](#) (pipeline). Небольшое напоминание о работе конвейера:

`cat file.txt | grep bmstu | sort -u`

`cat file.txt` - вывести содержимое файла, передаем это утилите grep

`grep bmstu` - оставить только строки, содержащие bmstu, передать вывод утилите sort

`sort -u` - отсортировать в алфавитном порядке и оставить только уникальные значения

Shell/Bash.

Пример скрипта, который создает файл с содержимым первого и второго аргументов:

```
#!/bin/sh

echo "$1" > test.txt
echo "$2" >> test.txt
```

Пример того, как сделать файл "исполняемым":

```
chmod +x file.sh
```

Пример того, как запустить "исполняемый" файл с аргументами из текущей директории:

```
./file.sh 10 15
```

Интересный факт: а почему исполняемый файл нельзя запустить просто написав file.sh как в windows?

Все дело в безопасности. Это очень плохая идея позволять подменять стандартные утилиты файлами из текущей директории, есть риск исполнить не то, что подразумевал пользователь, а то, что подложил ему в каталог злоумышленник. Поэтому текущая директория **не добавляется** к переменной PATH, списку директорий в которых производится поиск исполняемых файлов.

Man

Продемонстрировать умение пользоваться утилитой [man][<https://ru.wikipedia.org/wiki/Man>]. Вывести страницу мануала для утилиты apt.

Демонстрация проделанной работы происходит через вывод утилиты `history`.

Часть 2. Продвинутая работа с системой

Основная задача: научиться управлять процессами и создавать простейшую автоматизацию

2.1. Автоматизируем сбор данных о системе с bash

1. Выведем `LA`:

```
cat /proc/loadavg | awk '{ print $1,$2,$3" processes: "$4, last PID: "$5 }'
```

LA, Load average - среднее значение загрузки системы за некоторый период времени, как правило, отображается в виде трёх значений, которые представляют собой усредненные сглаженные величины за последние 1, 5 и 15 минут. Вычисляется как длина очереди выполнения в операционной системе, где единица означает, что очередь заполнена, а значение выше единицы — что есть процессы, которые ожидают своей очереди на выполнение. Содержится в файле `/proc/loadavg`. Учитывает как running-процессы, так и процессы в uninterruptable sleep (ожидают выполнения системного вызова, как правило долгие это read/write, т.е. IO).

2. Выведем список сетевых интерфейсов в нужном формате

```
ss -4tuln | подставьте что нужно, можно цепочку из нескольких команд |  
column -t
```

```
127.0.0.53%lo 53  
127.0.0.53%lo 53  
0.0.0.0 22
```

3. Выведем список IP-адресов на хосте:

```
ip a | grep -E 'inet' | awk '{ print $2 }'
```

4. Обернем это все в bash-скрипт /root/beholder.sh, добавим date +"%d.%m.%Y %H:%M:%S" и информацию о занятом месте и списке пользователей (самостоятельно). Попробуем циклы, запишем в скрипт:

```
for i in $@  
do  
    echo "Next search by name:"  
    ps aux | grep $i  
    echo  
done
```

5. Запустим. Должны получить что-то вроде:

```
08.09.2022 11:13:35  
  
LA: 0.00 0.00 0.00 processes: 1/132, last PID: 3194  
  
Listening ports:  
127.0.0.53%lo 53  
127.0.0.53%lo 53  
0.0.0.0 22  
  
IP-addresses:  
127.0.0.1/8  
10.0.2.15/24  
  
Disk free:  
Filesystem Size Used Avail Use% Mounted on  
udev 445M 0 445M 0% /dev  
tmpfs 98M 1.1M 97M 2% /run  
/dev/mapper/ubuntu--vg-ubuntu--lv 19G 4.4G 14G 25% /  
tmpfs 489M 0 489M 0% /dev/shm  
tmpfs 5.0M 0 5.0M 0% /run/lock  
tmpfs 489M 0 489M 0%  
/sys/fs/cgroup
```

```

/dev/loop0                      62M   62M    0 100%
/snap/core20/1611
/dev/loop1                      47M   47M    0 100%
/snap/snapd/16292
/dev/loop2                      68M   68M    0 100%
/snap/lxd/22753
/dev/mapper/ubuntu--vg-ubuntu--lv--home 9.8G   80K   9.3G   1% /home
/dev/sda2                        2.0G  106M   1.7G   6% /boot
tmpfs                           98M    0     98M   0%
/run/user/1000

Inode free:
Filesystem                  Inodes IUsed IFree IUse% Mounted on
udev                         112K   490   111K   1% /dev
tmpfs                        123K   768   122K   1% /run
/dev/mapper/ubuntu--vg-ubuntu--lv 1.2M   95K   1.1M   8% /
tmpfs                        123K    4    123K   1% /dev/shm
tmpfs                        123K    3    123K   1% /run/lock
tmpfs                        123K   18    123K   1%
/sys/fs/cgroup
/dev/loop0                     12K    12K    0 100%
/snap/core20/1611
/dev/loop1                     486   486    0 100%
/snap/snapd/16292
/dev/loop2                     802   802    0 100%
/snap/lxd/22753
/dev/mapper/ubuntu--vg-ubuntu--lv--home 640K   28    640K   1% /home
/dev/sda2                      128K   312   128K   1% /boot
tmpfs                          123K   22    123K   1%
/run/user/1000

Next search by name:
root      724  0.0  0.7  12172  7456 ?          Ss   06:50  0:00 sshd:
/usr/sbin/sshd -D [listener] 0 of 10-100 startups
root      1371  0.0  0.9  13952  9028 ?          Ss   06:56  0:00 sshd:
mak [priv]
mak      1453  0.0  0.5  13952  5860 ?          S    06:56  0:00 sshd:
mak@pts/1
mak      3192  0.0  0.3   7024  3556 pts/1      S+   10:49  0:00
/bin/bash ./beholder.sh sshd
mak      3206  0.0  0.0   6300   720 pts/1      S+   10:49  0:00 grep
sshd
=====
```

2.2. Делаем сбор данных регулярным с cron

8. Впишем в cron новое правило и ждем минуту.

```
echo '* * * * * root /root/beholder.sh >> /tmp/beholder-output' >
/etc/cron.d/beholder
```

9. Смотрим файл `cat /tmp/beholder-output`

Можем попробовать поменять параметры, сервис [crontab.guru](#) в помощь.

Краткое описание есть в файле конфига `/etc/crontab`:

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# ----- minute (0 - 59)
# | ----- hour (0 - 23)
# | | ----- day of month (1 - 31)
# | | | ----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | ----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | |
# * * * * * user-name command to be executed
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

Исходя из этого конфига следует, что в `/etc/cron.{hourly,daily,weekly,monthly}/` лежат скрипты, запускающиеся соответственно раз в день, в неделю и в месяц. Дополнительно Cron читает конфиги из `/etc/cron.d/`, где лежат общие файлы с расписанием запуска.

Кроме того, для пользователей задаются персональные cron-конфиги. Их можно отредактировать с помощью утилиты `crontab -e`. Они отличаются от `/etc/crontab` тем, что там опускается поле пользователя, запускаться будет от того, который себе задал это в настройках.

2.3. Пишем свой systemd-сервис

Процесс не может взяться из ниоткуда: его обязательно должен запустить какой-то процесс. Процесс, запущенный другим процессом, называется дочерним (child) процессом или потомком. Процесс, который запустил процесс называется родительским (parent), родителем или просто - предком. У каждого процесса есть два атрибута - **PID** (Process ID) - идентификатор процесса и **PPID** (Parent Process ID) - идентификатор родительского процесса.

Процессы создают иерархию в виде дерева. Самым "главным" предком, то есть процессом, стоящим на вершине этого дерева, является процесс systemd или init (PID = 1). Условно можно считать, что планировщик ядра имеет PID = 0.

Собственно для управления древом процессов и создали systemd. Подробнее о процессах и их состояниях читайте в [статье](#).

[**systemd**](#) — подсистема инициализации и управления службами в GNU/Linux, фактически вытеснившая в 2010-е годы традиционную подсистему [**init**](#). Основная особенность — интенсивное распараллеливание запуска служб в процессе загрузки системы, что позволяет существенно ускорить запуск операционной системы. Основная единица управления — unit (модуль), одним из типов модулей являются «сервисы» — аналог демонов — наборы процессов, запускаемые и управляемые средствами подсистемы и изолируемые [**контрольными группами**](#). Подробная [статья](#) об `systemd-unit-ax`.

Создадим свой `systemd-unit`:

```
nano /etc/systemd/system/myhttp.service

# Создадим для него WorkingDirectory
mkdir -p /root/www

# Перезагрузим конфигурацию systemd
systemctl daemon-reload
```

Содержимое `.service` файла:

```
[Unit]
Description=MyHTTP Server
Documentation=http://example.org/
After=network.target

[Service]
WorkingDirectory=/root/www
ExecStart=/usr/bin/python3 -m http.server 8000
KillMode=mixed
Restart=on-failure
Type=simple

[Install]
WantedBy=multi-user.target
Alias=myhttpd.service
```

Кратко разберем что тут написано:

Секция [Unit]:

`Description` - описание, отображается в `systemctl status [service]`

`Documentation` - ссылка на документацию, обязательно в виде URL

`After` - после каких целей/сервисов следует запускать данный unit

Секция [Service]:

`WorkingDirectory` - вызовет системный вызов `chdir` перед исполнением каких-либо команд сервиса, должна существовать

`ExecStart` - команда, которая запустится при вызове `systemctl start [service]`

`KillMode` - управляет тем, как процесс должен быть завершен, `mixed` - посыпает SIGTERM основному процессу и SIGKILL дочерним

Restart - необходимо ли перезапускать сервис в случае завершения процесса (в примере - только при ненулевом rc)

Type - настраивает процедуру запуска и отслеживания процесса, simple ожидает, что основным станет процесс, запущенный ExecStart командой и сервис будет работать до тех пор, пока жив основной процесс.

Секция [Install]:

WantedBy - какая цель триггерит запуск, если сервис enabled.

Alias - альтернативное имя сервиса. `systemctl status [alias]` равнозначен `systemctl start [service]`.

Запустим, добавим в автозапуск:

```
systemctl start myhttp
systemctl enable myhttp
```

Проверяем работу с помощью `curl` и `ss`.

```
curl http://127.0.0.1:8000/
ss -tl | grep 8000
```

Для ответа на контрольный вопрос про процесс загрузки системы также нужно ознакомиться с [материалом](#).

2.4. Запускаем процесс внутри нового пространства имен

Посмотрим список доступных пространств имен:

```
lsns
ip netns
```

Цикл [статей](#) на Habr про пространства имен.

Создаем сетевой namespace:

```
ip netns add myhttp
```

Теперь запустим внутри пару команд `ip netns exec <netns> <cmd> ...`:

```
ip netns exec myhttp ip link set dev lo up
ip netns exec myhttp /usr/bin/python3 -m http.server 8080 &
```

Проверяем:

```
ss -tul4n  
ip netns exec myhttp ss -tul4n
```

Запустим tcpdump внутри netns-а:

```
ip netns exec myhttp tcpdump -i any host 127.0.0.1
```

Проверим:

```
ip netns exec myhttp curl http://127.0.0.1:8080  
ip netns exec myhttp curl http://127.0.0.1:8000  
curl http://127.0.0.1:8080  
curl http://127.0.0.1:8000
```

Таким образом, у нас получилось разнести 2 разных сервера в разные сетевые пространства имен. Мы даже можем запустить их на одинаковом порту.

2.5. Установка docker и запуск hello-world

[Установка Docker в Ubuntu 20.04](#)

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common  
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -  
sudo add-apt-repository "deb [arch=amd64]  
https://download.docker.com/linux/ubuntu focal stable"  
sudo apt update  
sudo apt install docker-ce  
sudo systemctl status docker
```

Запустим [hello-world](#):

```
docker run hello-world
```

2.6. Дополнительное

1. Показать soft-лимиты (`ulimit`) и hard-лимиты, пояснить как поняли, в чем разница, как изменить?

Хорошая [статья](#) о лимитах.

Примечание: это может оказаться жизненно важным знанием в случае появления высоких нагрузок на вашем сервере. Ошибка "Too Many Open Files" может появляться как раз из-за ограничений в лимитах, по-умолчанию они довольно небольшие (1024 для open files).

Пример вывода `ulimit` для пары систем (ВМ и реальный Raspberry-Pi сервер):

```
make@enterprise:~$ uname -a
Linux enterprise 5.10.0-11-amd64 #1 SMP Debian 5.10.92-1 (2022-01-18) x86_64 GNU/Linux
make@enterprise:~$ ulimit -aS
real-time non-blocking time (microseconds, -R) unlimited
core file size (blocks, -c) 0
data seg size (kbytes, -d) unlimited
scheduling priority (-e) 0
file size (blocks, -f) unlimited
pending signals (-i) 3767
max locked memory (kbytes, -l) 124986
max memory size (kbytes, -m) unlimited
open files (-n) 1024
pipe size (512 bytes, -p) 8
POSIX message queues (bytes, -q) 819200
real-time priority (-r) 0
stack size (kbytes, -s) 8192
cpu time (Seconds, -t) unlimited
max user processes (-u) 3767
virtual memory (kbytes, -v) unlimited
file locks (-x) unlimited
make@enterprise:~$ ulimit -H
real-time non-blocking time (microseconds, -R) unlimited
core file size (blocks, -c) unlimited
data seg size (kbytes, -d) unlimited
scheduling priority (-e) 0
file size (blocks, -f) unlimited
pending signals (-i) 3767
max locked memory (kbytes, -l) 124986
max memory size (kbytes, -m) unlimited
open files (-n) 1048576
pipe size (512 bytes, -p) 8
POSIX message queues (bytes, -q) 819200
real-time priority (-r) 0
stack size (kbytes, -s) unlimited
cpu time (Seconds, -t) unlimited
max user processes (-u) 3767
virtual memory (kbytes, -v) unlimited
file locks (-x) unlimited
make@enterprise:~$ 
make@asuna:~$ uname -a
Linux asuna 5.10.17-v7l+ #1403 SMP Mon Feb 22 11:33:35 GMT 2021 armv7l GNU/Linux
make@asuna:~$ ulimit -aS
core file size (blocks, -c) 0
data seg size (kbytes, -d) unlimited
scheduling priority (-e) 0
file size (blocks, -f) unlimited
pending signals (-i) 60938
max locked memory (kbytes, -l) 64
max memory size (kbytes, -m) unlimited
open files (-n) 1024
pipe size (512 bytes, -p) 8
POSIX message queues (bytes, -q) 819200
real-time priority (-r) 0
stack size (kbytes, -s) 8192
cpu time (Seconds, -t) unlimited
max user processes (-u) 60938
virtual memory (kbytes, -v) unlimited
file locks (-x) unlimited
make@asuna:~$ ulimit -H
core file size (blocks, -c) unlimited
data seg size (kbytes, -d) unlimited
scheduling priority (-e) 0
file size (blocks, -f) unlimited
pending signals (-i) 60938
max locked memory (kbytes, -l) 64
max memory size (kbytes, -m) unlimited
open files (-n) 524288
pipe size (512 bytes, -p) 8
POSIX message queues (bytes, -q) 819200
real-time priority (-r) 0
stack size (kbytes, -s) unlimited
cpu time (Seconds, -t) unlimited
max user processes (-u) 60938
virtual memory (kbytes, -v) unlimited
file locks (-x) unlimited
make@asuna:~$ 
```

Посмотрим лимиты для конкретного процесса, тоже бывает полезно:

```
# Некоторые программы кладут свой Process ID (PID) в /var/run
cat /var/run/sshd.pid # Выведет PID sshd

# Выведем лимиты процесса sshd с помощью procfs
cat /proc/$(cat /var/run/sshd.pid)/limits
# P.S. Про procfs подробно поговорим во второй лабораторной работе
```

Limit	Soft Limit	Hard Limit	Units
Max cpu time	unlimited	unlimited	seconds
Max file size	unlimited	unlimited	bytes
Max data size	unlimited	unlimited	bytes
Max stack size	8388608	unlimited	bytes
Max core file size	0	unlimited	bytes
Max resident set	unlimited	unlimited	bytes
Max processes	3767	3767	processes
Max open files	1024	524288	files
Max locked memory	65536	65536	bytes
Max address space	unlimited	unlimited	bytes
Max file locks	unlimited	unlimited	locks
Max pending signals	3767	3767	signals
Max msgqueue size	819200	819200	bytes
Max nice priority	0	0	
Max realtime priority	0	0	
Max realtime timeout	unlimited	unlimited	us

Занятное наблюдение работы лимитов:

```

[mak@asuna:~/files $ ulimit -Sn # смотрим soft-лимит open files
1024
[mak@asuna:~/files $ ulimit -Hn # смотрим hard-лимит open files
524288
[mak@asuna:~/files $ for i in {0..2000}; do touch ${i}.txt; done
[mak@asuna:~/files $ # Создали 2001 пустых файлов
[mak@asuna:~/files $
[mak@asuna:~/files $ cat check.py
import sys

if len(sys.argv) < 2:
    print("Input file count as first argument")
    exit(1)

fds = list()
try:
    for i in range(int(sys.argv[1])):
        f = open(str(i) + ".txt")
        fds.append(f)
except Exception as err:
    print("Error:", err)

print("Open files (not actually all fd): ", len(fds))
[mak@asuna:~/files $
[mak@asuna:~/files $ python check.py 100
('Open files (not actually all fd): ', 100)
[mak@asuna:~/files $ python check.py 1024
('Error:', IOError(24, 'Too many open files'))
('Open files (not actually all fd): ', 1021)
[mak@asuna:~/files $
[mak@asuna:~/files $ ulimit -Sn 2048 # меняем soft-лимит open files
[mak@asuna:~/files $ python check.py 1024
('Open files (not actually all fd): ', 1024)
[mak@asuna:~/files $ ulimit -Sn # смотрим soft-лимит open files
2048
[mak@asuna:~/files $ ulimit -Hn # смотрим hard-лимит open files
524288
[mak@asuna:~/files $ ]

```

2. Настроить `/root/.bashrc` файл, применяющийся при создании новой оболочки для root пользователя.

Хорошая [статья](#) о bash_profile и bashrc.

[Пример](#) большого и сложного .bashrc файла.

PS1 - переменная для изменения вида приглашения, например такое значение красит в красный:

```

PS1='[\e]0;\u@\h: \w\a\$${debian_chroot:+($debian_chroot)}\
[\e[01;31m]\u@\h[\e[00m]:[\e[01;34m]\w \$[\e[00m] '

```

Синонимы (alias-ы) для подкрашивания вывода команд:

```
# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval
"$(dircolors -b)"
    alias ls='ls --color=auto'
    alias dir='dir --color=auto'
    alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi
```

Просто удобные общепринятые сокращения:

```
# some more ls aliases
alias ll='ls -l'
alias la='ls -A'
alias l='ls -lA'
```

Переопределения стандартных команд с ключами, чтобы спрашивали подтверждение:

```
# Some more alias to avoid making mistakes:
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
```

```

GNU nano 5.4                                .bashrc
# ~/.bashrc: executed by bash(1) for non-login shells.

# Note: PS1 and umask are already set in /etc/profile. You should not
# need this unless you want different defaults for root.
# PS1='${debian_chroot:+($debian_chroot)}\h:\w\$ '
# umask 022
PS1='[\e[0;32m$debian_chroot:+($debian_chroot)]\[\033[01;31m\]\w@\h\[\033[0m]'

# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval "$(dircolors -b)"
    alias ls='ls --color=auto'
    alias dir='dir --color=auto'
    alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi

# some more ls aliases
alias ll='ls -l'
alias la='ls -A'
alias l='ls -la'

# Some more alias to avoid making mistakes:
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'

[mak@enterprise:~$ su -
[Password:
[root@enterprise:~ # # Cool!
[root@enterprise:~ #
logout
mak@enterprise:~$ ]

```

3. Создайте еще один мгновенный снимок ВМ, как в пункте 6 раздела 1.1.
Чтобы не случалось “ups, у меня все работало, а вот сейчас перестало”.

Далее в лабораторных рекомендуется использовать способ работы с системой по ssh и запускать ВМ в фоновом режиме (без интерфейса вообще)

Контрольные вопросы

1. Что такое Linux?
2. Что делает каждая из команд, применяемых в лабораторной работе?
3. В чем разница между su и sudo?
4. Что происходит в подробностях, когда вы вводите `cat file.txt`?
5. Что такое файловый дескриптор? Какие создаются по умолчанию?
6. Как дописать в файл вывод команды? Как перезаписать файл? Как избавиться от вывода ошибок?

7. Как определяются права на файл? Как сделать файл исполняемым?
8. Как создать пользователя в GNU/Linux?
9. Как пользоваться man?
10. Как выдать права на sudo?
11. Зачем .bashrc файл?
12. Как посмотреть занятое место на диске?
13. Как подключиться к серверу по ssh? Подробно о всех возможных проблемах.
14. Что такое публичный и приватный ключ? Какой оставляем себе, а какой копируем на сервер?
15. ssh-agent - зачем?
16. В чем разница между soft и hard лимитами?
17. Кто такой суперпользователь и что ему можно?
18. Нужно ли делать резервные копии? А что еще нужно делать?
19. Что такое systemd, как происходит загрузка системы?
20. Как поставить пакет? Что такое apt?

Для выбора вопроса можно использовать игральную кость d20