Linux

Базы данных

Установка и первоначальная настройка MySQL (Debian/Ubuntu)

Установка сервера баз данных MySQL:

```
sudo apt install mysql-server
```

Вход в интерфейс управления MySQL при помощи пользователя root

```
mysql -u root
```

Установка пароля на пользователя root

```
USE mysql;
ALTER USER 'root'@'localhost' IDENTIFIED BY 'password';
```

Создание базы данных MySQL и её пользователя (Debian/Ubuntu)

Создание базы данных

```
CREATE DATABASE DataBaseName;
```

Создание пользователя

```
CREATE USER 'user_name'@'%' IDENTIFIED WITH mysql_native_password BY 'password';
```

Настройка пользователю всех прав на базу данных

```
GRANT ALL PRIVILEGES ON DataBaseName . * TO 'user_name'@'%';
```

Установка и первоначальная настройка MariaDB (Debian/Ubuntu)

Установка сервера и клиентские утилиты баз данных MariaDB:

```
sudo apt install mariadb-server && sudo apt install mariadb-client
```

Запуск утилиты настройки сервера баз данных:

```
mysql_secure_installation
```

Далее утилита попросит ввести пароль для root, после утилита спросит: Switch to unix_socket authentication, выбираем ответ "y".

Следующие вопросы и ответы на них:

Change the root password? Т.к пароль на root уже был установлен – выбираем ответ "n".

Remove anonymous users? Отвечаем "y".

Disallow root login remotely? Отключаем удаленный доступ к руту, следовательно, выбираем ответ "y".

Remove test database and access to it? Удаляем тестовую базу данных, выбираем ответ "y".

Reload privilege tables now? Перезагружаем таблицу привилегий, выбираем ответ "y".

В файле /etc/mysql/mariadb.conf.d/50-server.cnf можно поменять bind-address с 127.0.0.1 на 0.0.0.0, чтобы другие компьютеры в сети могли подключиться к серверу баз данных.

Создание базы данных MariaDB и её пользователя (Debian/Ubuntu)

Создание базы данных

```
CREATE DATABASE DataBaseName;
```

Создание пользователя

```
CREATE USER 'user_name'@'%' IDENTIFIED BY 'password';
```

Настройка пользователю всех прав на базу данных

```
GRANT ALL PRIVILEGES ON DataBaseName . * TO 'user_name'@'%';
```

LXC

Линукс контейнер

Первоначальная настройка контейнера

Настройка времени:

Установка часового пояса

```
sudo timedatectl set-timezone Europe/Moscow
```

В качестве примера указан часовой пояс Москвы, но разумеется нужно указать свой, если он конечно не совпадает с московским.

Добавление SSH сертификата

```
ssh-copy-id -i ~/.ssh/файл_ключа.pub имя_пользователя@хост
```

LXC

Установка поддержки русского языка (Ubuntu)

```
sudo apt install language-pack-ru language-pack-ru-base manpages && dpkg-reconfigure locales
```

Выбираем ru_RU.UTF-8 и перезагружаем контейнер.

LXC

Установка поддержки русского языка (Debian)

```
sudo dpkg-reconfigure locales
```

Выбираем ru_RU.UTF-8 UTF-8, на следующем этапе оставляем C.UTF-8 в качестве локализации по умолчанию, если не хотим, чтобы интерфейс системы стал на русском языке, либо выбираем ru_RU.UTF-8 если это необходимо.

LXC

Уменьшение размера виртуального диска контейнера

> За основу взята следующая статья

Сначала необходимо остановить контейнер и сделать бекап, и проверить что он есть.

Следующие действия не работают с виртуальными дисками на LVM носителями, следует временно перенести их на какой-нибудь диск с файловой системой ext4.

Далее следует зайти в каталог с файлом диска, это может быть к примеру /mnt/usb_hdd/images

Проверка файловой системы:

```
e2fsck -f vm-100-disk-0.raw
```

(Имя файла vm-100-disk-0.raw условно, у вас оно может быть другим)

Если утилита выдает следующее:

> e2fsck: MMP: fsck being run while checking MMP block
> MMP check failed: If you are sure the filesystem is not in use on any node, run:
> ‘tune2fs -f -E clear_mmp {device}’

следует выполнить следующее:

```
tune2fs -f -E clear_mmp vm-100-disk-0.raw
```

После чего должен быть примерно следующий результат:

```
e2fsck 1.47.0 (5-Feb-2023)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
vm-100-disk-1.raw: 26392/1638400 files (0.2% non-contiguous),
353303/6553600 blocks
```

Далее следует уменьшить размер структуры файловой системы в файле диска:

```
resize2fs -M vm-100-disk-0.raw 4G
```

где 4G - желаемый размер диска

> Если объем файлов превышает вводимое значение, то образ диска будет повреждён!

Далее следует обрезать лишнюю часть образа, до размера указанного в предыдущем шаге:

```
truncate -s 4G vm-100-disk-0.raw
```

Далее расширяем файловую структуру до размеров виртуального диска:

```
resize2fs vm-100-disk-0.raw
```

После следует изменить значение размера образа диска в настройках контейнера:

```
nano /etc/pve/nodes/pve1/lxc/100.conf
```

где pve1 - имя ноды Proxmox VE, 100.conf - конфигурация контейнера с id 100.

Изменяем параметр size:

rootfs: local:100/vm-100-disk-0.raw,size=**4G**

Сохраняем изменения и запускаем контейнер.

Посмотреть изменения можно подключившись к контейнеру по SSH и ввести команду:

```
df -h
```

Веб-сервер Apache

Установка Apache+PHP-FMP 8.2 (Debian/Ubuntu)

Установка Apache:

```
sudo apt install apache2
```

Редактирование опции <Directory /var/www/> в /etc/apache2/apache2.conf:

```
<Directory /var/www/>
        Options Indexes ExecCGI FollowSymLinks
        AllowOverride All
        Require all granted
</Directory>
```

Установка пакетов PHP, в данном случае используется версия 8.2, но можно использовать любую другую по желанию, просто замените циферки 8.2 на нужные вам:

```
sudo apt install php-fpm php-gd php-mbstring php-mysql php-zip php-memcached memcached
```

Включение нужных сервисов, PHP-FPM и кэширования:

```
sudo systemctl enable php8.2-fpm && sudo systemctl enable memcached
```

Замена модуля mpm_event на mpm_prefork:

```
sudo a2dismod mpm_event && sudo a2enmod mpm_prefork
```

Активация модуля PHP-FPM:

```
sudo a2enconf php8.2-fpm
```

Установка библиотеки libapache2-mod-fcgid для обмена данными для Apache HTTP и PHP:

```
sudo apt install libapache2-mod-fcgid
```

Активация модуля proxy и proxy_fcgi:

```
sudo a2enmod proxy && sudo a2enmod proxy_fcgi
```

Добавим в конфигурационный файл /etc/apache2/mods-available/dir.conf index.php в самое начало, должно получиться

> DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm

Проверка конфигурации:

```
sudo apachectl configtest
```

> Output
>
> Syntax OK

Для установки часового пояса в настройках PHP следует найти параметр date.timezone, раскомментировать и указать значение:

```
sudo nano /etc/php/8.2/fpm/php.ini
```

Перезапуск PHP-FPM и Apache:

```
sudo systemctl restart php8.2-fpm && sudo systemctl restart apache2
```

Создание виртуального хоста (сайта)

Пример виртуального хоста:

```
<VirtualHost *:80>
	ServerName example.com
	
	UseCanonicalName On

	DocumentRoot /var/www/example_com

	<Directory /var/www/example_com>
		Options FollowSymLinks
		AllowOverride All
		Require all granted
	</Directory>

	ErrorLog ${APACHE_LOG_DIR}/example_com-error.log
	CustomLog ${APACHE_LOG_DIR}/example_com-access.log combined

</VirtualHost>
```

Центр сертификации

> Подробнее (Оригинальная статья): здесь

Установка Easy-RSA

```
sudo apt update
sudo apt install easy-rsa
```

Подготовка директории для инфраструктуры открытых ключей

```
mkdir ~/easy-rsa
```

Создайте символические ссылки с помощью команды `ln`:

```
ln -s /usr/share/easy-rsa/* ~/easy-rsa/
```

Чтобы ограничить доступ к созданной директории PKI, используйте команду `chmod` для предоставления доступа к ней только владельцу:

```
chmod 700 /home/sammy/easy-rsa
```

Затем инициализируйте PKI в директории `easy-rsa`:

```
cd ~/easy-rsa
./easyrsa init-pki
```

Output

```
init-pki complete; you may now create a CA or requests.
Your newly created PKI dir is: /home/sammy/easy-rsa/pki
```

Создание Центра сертификации

```
cd ~/easy-rsa
nano vars
```

Открыв файл, вставьте следующие строки и измените каждое выделенное значение для отражения информации о вашей организации. При этом важно, чтобы ни одно значение не

оставалось пустым:

~/easy-rsa/vars

```
set_var EASYRSA_REQ_COUNTRY    "US"
set_var EASYRSA_REQ_PROVINCE   "NewYork"
set_var EASYRSA_REQ_CITY       "New York City"
set_var EASYRSA_REQ_ORG        "DigitalOcean"
set_var EASYRSA_REQ_EMAIL      "admin@example.com"
set_var EASYRSA_REQ_OU         "Community"
set_var EASYRSA_ALGO           "ec"
set_var EASYRSA_DIGEST         "sha512"
```

Для создания корневой пары открытого и закрытого ключей для Центра сертификации необходимо запустить команду `./easy-rsa` еще раз, но уже с опцией `build-ca`:

```
./easyrsa build-ca
```

Output

```
. . .
Enter New CA Key Passphrase:
Re-Enter New CA Key Passphrase:
. . .
Common Name (eg: your user, host, or server name) [Easy-RSA CA]:

CA creation complete and you may now import and sign cert requests.
Your new CA certificate file for publishing is at:
/home/sammy/easy-rsa/pki/ca.crt
```

> Если вы не хотите вводить пароль при каждом взаимодействии с ЦС, вы можете запустить команду `build-ca` с опцией `nopass`:

```
./easyrsa build-ca nopass
```

Распространение публичного сертификата Центра сертификации

Чтобы импортировать публичный сертификат ЦС во вторую систему Linux, например на сервер или локальный компьютер, нужно предварительно получить копию файла `ca.crt` с сервера ЦС. Вы можете использовать команду `cat` для ее вывода в терминал, а затем скопировать и вставить ее в файл на втором компьютере, который импортирует сертификат. Также вы можете использовать `scp`, `rsync` и другие подобные инструменты для передачи файла между системами. Мы используем для копирования и вставки текстовый редактор

`nano`, поскольку этот вариант подойдет для всех систем.

Запустите следующую команду на сервере ЦС от имени пользователя без прав root:

```
cat ~/easy-rsa/pki/ca.crt
```

На терминале появится примерно следующее:

Output

```
-----BEGIN CERTIFICATE-----
MIIDSzCCAjOgAwIBAgIUcR9Crsv3FBEujrPZnZnU4nSb5TMwDQYJKoZIhvcNAQEL
BQAwFjEUMBIGA1UEAwwLRWFzeS1SU0EgQ0EwHhcNMjAwMzE4MDMxNjI2WhcNMzAw
. . .
. . .
-----END CERTIFICATE-----
```

Скопируйте все, включая строки `-----BEGIN CERTIFICATE-----` и `-----END CERTIFICATE-----` и символы дефиса.

Используйте `nano` или предпочитаемый текстовый редактор на второй системе Linux, чтобы открыть файл с именем `/tmp/ca.crt`:

```
nano /tmp/ca.crt
```

Теперь у нас имеется копия файла `ca.crt` на второй системе Linux, и мы можем импортировать сертификат в хранилище сертификатов операционной системы.

На системах с Ubuntu и Debian выполните следующие команды в качестве вашего пользователя без прав root для импорта сертификата:

```
sudo cp /tmp/ca.crt /usr/local/share/ca-certificates/
sudo update-ca-certificates
```

Чтобы импортировать сертификат сервера ЦС в систему на базе CentOS, Fedora или RedHat, скопируйте и вставьте содержимое файла в файл `/tmp/ca.crt`, как описано в предыдущем примере. Затем скопируйте сертификат в директорию `/etc/pki/ca-trust/source/anchors/` и запустите команду `update-ca-trust`.

```
sudo cp /tmp/ca.crt /etc/pki/ca-trust/source/anchors/
sudo update-ca-trust
```

Теперь вторая система Linux будет доверять любому сертификату, подписанному нашим сервером ЦС.

> Если вы используете ЦС с веб-серверами и браузер Firefox, вам нужно будет импортировать публичный сертификат `ca.crt` в Firefox напрямую. Firefox не использует локальное хранилище сертификатов операционной системы. Подробную информацию о добавлении сертификата ЦС в Firefox можно найти в статье поддержки Mozilla «Настройка Центров сертификации (ЦС) в Firefox».

> Если вы используете ЦС для интеграции со средой Windows или настольными компьютерами, ознакомьтесь с документацией по использованию `certutil.exe` для установки сертификата ЦС.

Создание запросов на подписание сертификатов и отзыв сертификатов

Создание и подписание образца запроса сертификата

`openssl` обычно устанавливается по умолчанию в большинстве дистрибутивов Linux, но для уверенности стоит запустить в системе следующую команду:

```
sudo apt update
sudo apt install openssl
```

В первую очередь для создания CSR необходимо сгенерировать закрытый ключ. Чтобы создать закрытый ключ с помощью `openssl`, создайте директорию `practice-csr` и сгенерируйте ключ в этой директории. Мы будем выполнять этот запрос на фиктивном сервере под названием `sammy-server`, в отличие от случая создания сертификата для идентификации пользователя или другого ЦС.

```
mkdir ~/practice-csr
cd ~/practice-csr
openssl genrsa -out sammy-server.key
```

Output

```
Generating RSA private key, 2048 bit long modulus (2 primes)
. . .
. . .
e is 65537 (0x010001)
```

Теперь у нас имеется закрытый ключ, с помощью которого можно создать CSR, используя утилиту `openssl`. Вам будет предложено заполнить ряд полей, в том числе указать страну, область и город. Вы можете ввести `.`, если хотите оставить поле пустым, но для реальных

CSR лучше использовать правильные значения при указании своего расположения и организации:

```
openssl req -new -key sammy-server.key -out sammy-server.req
```

Output

```
. . .
-----
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:New York
Locality Name (eg, city) [Default City]:New York City
Organization Name (eg, company) [Default Company Ltd]:DigitalOcean
Organizational Unit Name (eg, section) []:Community
Common Name (eg, your name or your server's hostname) []:sammy-server
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Когда вас устроит тема запроса тренировочного сертификата, скопируйте файл `sammy-server.req` на сервер ЦС с помощью `scp`:

```
scp sammy-server.req sammy@your_ca_server_ip:/tmp/sammy-server.req
```

Подписание CSR

Первым шагом для подписания вымышленного CSR будет импорт запроса сертификата с помощью скрипта `easy-rsa`:

```
cd ~/easy-rsa
./easyrsa import-req /tmp/sammy-server.req sammy-server
```

Output

```
. . .
The request has been successfully imported with a short name of: sammy-server
You may now use this name to perform signing operations on this request.
```

Теперь вы можете подписать запрос, запустив скрипт `easyrsa` с опцией `sign-req`, указав затем тип запроса и общее имя, включаемое в CSR. Запрос может иметь тип `client`, `server` или `ca`.

Поскольку мы тренируемся с сертификатом для вымышленного сервера, нужно использовать тип запроса `server`:

```
./easyrsa sign-req server sammy-server
```

В результатах вам будет предложено подтвердить, что запрос поступил из надежного источника. Для подтверждения введите `yes` и нажмите `ENTER`:

Output

```
You are about to sign the following certificate.
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.

Request subject, to be signed as a server certificate for 3650 days:

subject=
    commonName                = sammy-server



Type the word 'yes' to continue, or any other input to abort.
  Confirm request details: yes
. . .
Certificate created at: /home/sammy/easy-rsa/pki/issued/sammy-server.crt
```

Если вы зашифровали ключ ЦС, вам будет предложено ввести пароль.

Выполнив эти шаги, мы подписали CSR `sammy-server.req` с помощью закрытого ключа сервера ЦС в директории `/home/sammy/easy-rsa/pki/private/ca.key`. Полученный файл `sammy-server.crt` содержит открытый ключ шифрования тренировочного сервера, а также новую подпись от сервера ЦС. Подпись сообщает всем, кто доверяет ЦС, что они также могут доверять сертификату `sammy-server`.

Если бы это был запрос веб-сервера, сервера VPN или другого реального сервера, последним шагом на сервере ЦС стало бы распространение новых файлов `sammy-server.crt` и `ca.crt` с сервера ЦС на удаленный сервер, отправивший запрос CSR:

```
scp pki/issued/sammy-server.crt sammy@your_server_ip:/tmp
scp pki/ca.crt sammy@your_server_ip:/tmp
```

Отзыв сертификата

Иногда сертификат требуется отозвать, чтобы пользователь или сервер не могли его использовать. Например, это может потребоваться в случае кражи ноутбука, взлома веб-сервера, увольнения сотрудника, расторжения договора с подрядчиком и т. д.

Далее кратко описана процедура отзыва сертификата:

1. Для отзыва сертификата используется команда `./easyrsa revoke client_name`.
2. Сгенерируйте новый CRL с помощью команды `./easyrsa gen-crl`.
3. Переместите обновленный файл `crl.pem` на сервер или серверы, использующие ваш ЦС, а на этих системах скопируйте этот файл в директорию или директории программ, которые на него ссылаются.
4. Перезапустите все службы, использующие ваш ЦС и файл CRL.

С помощью этой процедуры вы можете отозвать любые сертификаты, которые ранее выпустили для вашего сервера. В следующих разделах мы подробно рассмотрим каждый шаг, начиная с команды `revoke`.

Для отзыва сертификата перейдите в директорию `easy-rsa` на вашем сервере ЦС:

```
cd ~/easy-rsa
```

Затем запустите скрипт `easyrsa` с опцией `revoke`, указав имя клиента, у которого хотите отозвать сертификат: В соответствии с приведенным выше практическим примером, сертификат имеет обычное имя `sammy-server`:

```
./easyrsa revoke sammy-server
```

Система предложит вам подтвердить отзыв сертификата. Введите `yes`:

```
Output
Please confirm you wish to revoke the certificate with the following subject:


subject=
    commonName                = sammy-server



Type the word 'yes' to continue, or any other input to abort.
  Continue with revocation: yes
. . .
Revoking Certificate 8348B3F146A765581946040D5C4D590A
. . .
```

Обратите внимание на выделенное значение в строке `Revoking Certificate`. Это значение представляет собой уникальный серийный номер отзываемого сертификата. Данное

значение потребуется вам, если вы захотите просмотреть список отзыва и убедиться в наличии в нем сертификата, как описано в последнем шаге этого раздела.

После подтверждения действия ЦС выполнит отзыв сертификата. Однако удаленные системы, использующие ЦС, не имеют возможности проверить отзыв сертификатов. Пользователи и серверы смогут использовать этот сертификат, пока список отзыва сертификатов ЦС (CRL) не будет распространен по всем системам, использующим данный ЦС.

На следующем шаге мы сгенерируем CRL или обновим существующий файл `crl.pem`.

Генерирование списка отзыва сертификатов

Мы отозвали сертификат, и теперь нам нужно обновить список отозванных сертификатов на сервере ЦС. После получения обновленного списка отзыва вы сможете определить, какие пользователи и системы имеют действующие сертификаты в вашем ЦС.

Чтобы сгенерировать CRL, запустите команду `easy-rsa` с опцией `gen-crl`, оставаясь в директории `~/easy-rsa`:

```
./easyrsa gen-crl
```

Если вы использовали фразу-пароль при создании файла `ca.key`, вам будет предложено ввести ее. Команда `gen-crl` сгенерирует файл с именем `crl.pem`, содержащий обновленный список отозванных сертификатов для этого ЦС.

Далее вам нужно будет передавать обновленный файл `crl.pem` на все серверы и клиенты, использующие этот ЦС, при каждом запуске команды `gen-crl`. В противном случае клиенты и системы сохранят доступ к сервисам и системам, использующим ваш ЦС, так как данным сервисам нужно сообщить об отзыве сертификата.

Передача списка отзыва сертификатов

Мы сгенерировали список CRL на сервере ЦС, и теперь нам нужно передать его на удаленные системы, использующие ваш ЦС. Для передачи этого файла на ваши серверы можно использовать команду `scp`.

> В этом обучающем руководстве описывается генерирование и распространение списка CRL вручную. Хотя существуют более надежные автоматические методы распространения и проверки списков отзыва (например OCSP-Stapling), настройка этих методов не входит в состав данного обучающего руководства.

Войдите на сервер ЦС как пользователь без прав root и запустите следующую команду, указав IP-адрес или имя DNS вашего сервера вместо `your_server_ip`:

```
scp ~/easy-rsa/pki/crl.pem sammy@your_server_ip:/tmp
```

Теперь файл передан на удаленную систему и нужно только отправить новую копию списка отзыва во все сервисы.

Сеть

Сеть

Создание программного роутера Debian

> За основу взята данная статья.

> Все действия выполняются на виртуальной машине в Proxmox VE.

Сначала требуется чтобы у виртуальной машины, которая будет выступать в роли роутера, было минимум два сетевых интерфейса, в данном случае они представляют собой два Linux Bridge, в качестве системы виртуализации используется Proxmox. Первый мост – стандартный vmbr0, на нем будет внешняя сеть роутера, второй мост vmbr1 – на нем будет внутренняя сеть роутера, в настройках сетевых устройств Proxmox настроен CIDR: 10.0.1.0/24

После установки Debian в виртуалку настраиваем сетевые интерфейсы:

В данном примере система имеет два сетевых интерфейса: enp6s18 и enp6s19.

enp6s18 – это vmbr0 и будет внешней сетью

enp6s19 – это vmbr1 и будет внутренней сетью.

Редактируем файл /etc/networking/interfaces, содержимое должно иметь примерно следующий вид:

```
auto enp6s18
iface enp6s18 inet static
address 192.168.1.100
netmask 255.255.255.0
gateway 192.168.1.1

auto enp6s19
iface enp6s19 inet static
address 10.0.1.1
netmask 255.255.255.0
```

Далее перезапускаем сетевую службу:

```
sudo systemctl restart networking.service
```

Настроим правила файрволла

Создадим каталог /etc/firewall и скрипт firewall.sh в нём:

```
sudo mkdir /etc/firewall
sudo nano /etc/firewall/iptables.sh
```

В нем следует создать следующее содержимое:

```
#!/bin/sh

sysctl -w net.ipv4.ip_forward=1

iptables -F
iptables -t nat -A POSTROUTING -o enp6s18 -j MASQUERADE
```

> Следует быть внимательным, в правилах указывается маска интерфейсов и она может отличаться!

Делаем скрипт исполняемым:

```
sudo chmod +x /etc/firewall/iptables.sh
```

Создадим службу rc.local

Создаем сервис для службы:

```
sudo nano /etc/systemd/system/rc-local.service
```

В него добавляем следующий код:

```
[Unit]
Description=/etc/rc.local
ConditionPathExists=/etc/rc.local
[Service]
Type=forking
ExecStart=/etc/rc.local start
```

```
TimeoutSec=0
StandardOutput=tty
RemainAfterExit=yes
SysVStartPriority=99
[Install]
WantedBy=multi-user.target
```

Создадим файл /etc/rc.local

```
sudo nano /etc/rc.local
```

Помещаем туда следующий код:

```
#!/bin/sh -e


/etc/firewall/iptables.sh
exit 0
```

Делаем его исполняемым:

```
sudo chmod +x /etc/rc.local
```

Устанавливаем dnsmasq

Устанавливаем:

```
sudo apt install dnsmasq
```

Открываем конфиг:

```
sudo nano /etc/dnsmasq.conf
```

Добавляем в конец файла следующее:

```
bind-interfaces
domain-needed
bogus-priv
interface=enp6s19
resolv-file=/etc/resolv.conf
dhcp-range=10.0.1.154,10.0.1.254,24h
cache-size=150
```

Этим самым мы установили и сконфигурировали DHCP-сервер и добавили диапазон из 100 адресов с 10.0.1.154 до 10.0.1.254 в качестве адресов, которые будут назначаться автоматически, значения можно поменять на свой вкус.

> В качестве интерфейса был назначен enp6s19, который соответствует локальной сети роутера и подключен к мосту vmbr1!

Перезагружаем виртуалку: sudo reboot now

Проверяем работу и радуемся.