OTUS

ОНЛАЙН-ОБРАЗОВАНИЕ

Онлайн-образование

Не забыл включить запись

Меня хорошо видно && слышно?

Ставьте плюсы, если все хорошо

Напишите в чат, если есть проблемы

Правила вебинара

- Активно участвуем
- Задаем вопросы в чат или голосом
- Off-topic обсуждаем в Slack #канал группы или #general
- Вопросы вижу в чате, могу ответить не сразу

Виртуализация: KVM

Маршрут вебинара

- QEMU-KVM
- Установка KVM
- Управление KVM

Цели занятия

После занятия вы сможете:

1. Познакомиться с гипервизором KVM
2. Понять чем KVM отличается от своих аналогов
3. Научиться разворачивать и администрировать гипервизор на основе KVM

Смысл

Зачем вам это уметь:

1. Чтобы понимать основные особенности использования QEMU-KVM
2. Чтобы наиболее эффективно использовать QEMU-KVM в вашей инфраструктуре
3. Чтобы сделать осознанный выбор наиболее подходящего инструмента для решения ваших задач

QEMU-KVM

KVM

Вопрос к аудитории: "Знакомы ли вы с KVM?"

KVM

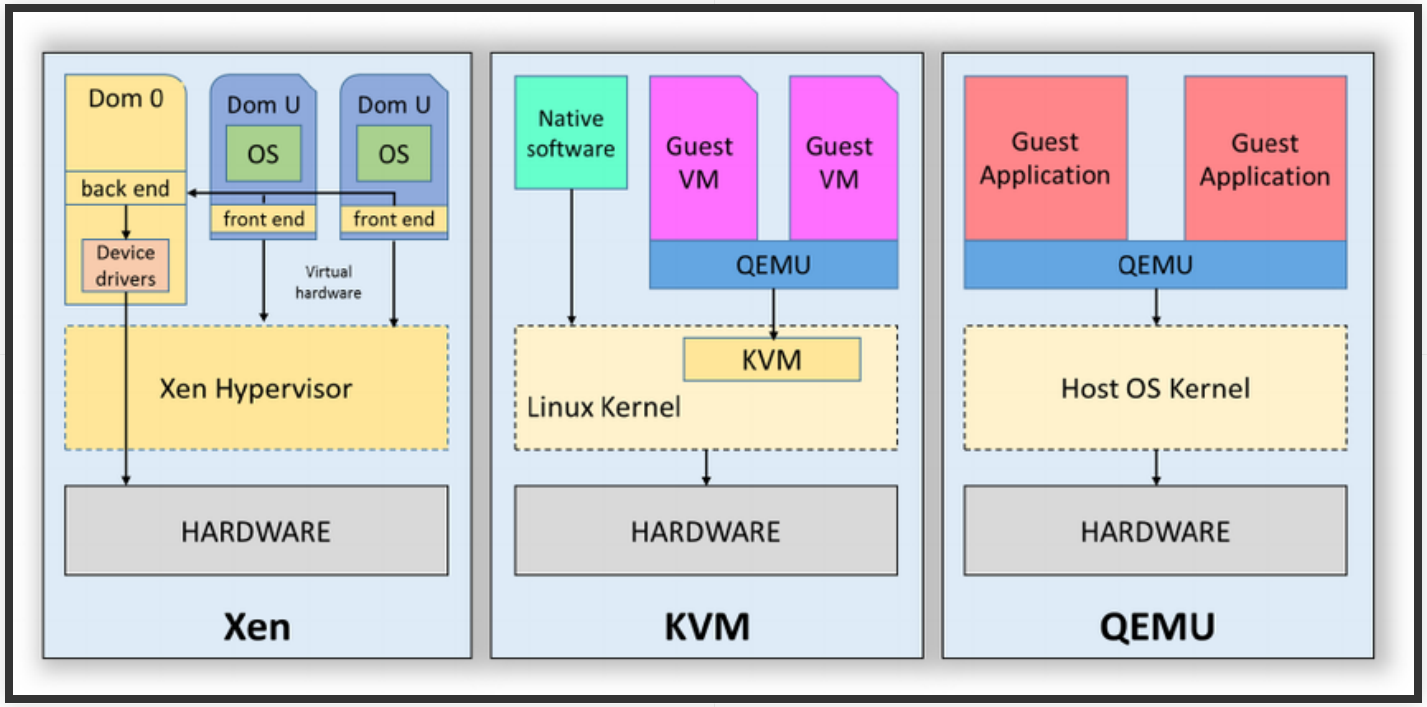
KVM - (Kernel-based Virtual Machine) — программное решение, обеспечивающее виртуализацию в среде Linux на платформе x86, которая поддерживает аппаратную виртуализацию на базе Intel VT (Virtualization Technology) либо AMD SVM (Secure Virtual Machine)

KVM

Особенности:

- Состоит по сути из модулей ядра
- Модуль kvm - базовая виртуализация
- Модули kvm-amd и kvm-intel - в зависимости от процессора
- KVM не выполняет эмуляцию, а использует эмулятор QEMU

KVM


Dom 0
Dom U
OS
Dom U
OS
back end
front end
front end
Device drivers
Virtual hardware
Xen Hypervisor
HARDWARE
Xen
Native software
Guest VM
Guest VM
QEMU
KVM
Linux Kernel
HARDWARE
KVM
Guest Application
Guest Application
QEMU
Host OS Kernel
HARDWARE
QEMU

QEMU

Вопрос к аудитории: "Зачем нужен QEMU?"

QEMU

QEMU — свободная программа с открытым исходным кодом для эмуляции аппаратного обеспечения различных платформ

QEMU

Особенности:

- Включает в себя эмуляцию процессоров Intel x86 и устройств ввода-вывода
- Может эмулировать 80386, 80486, Pentium, Pentium Pro, AMD64 и другие x86-совместимые процессоры; ARM, MIPS, RISC-V, PowerPC, SPARC, SPARC64 и частично m68k
- Эмулирует оборудование программно, как следствие - низкая производительность
- Для подключения аппаратной эмуляции использует KVM

QEMU пример эмуляции

Основной конфиг QEMU - **/etc/libvirt/qemu.conf**

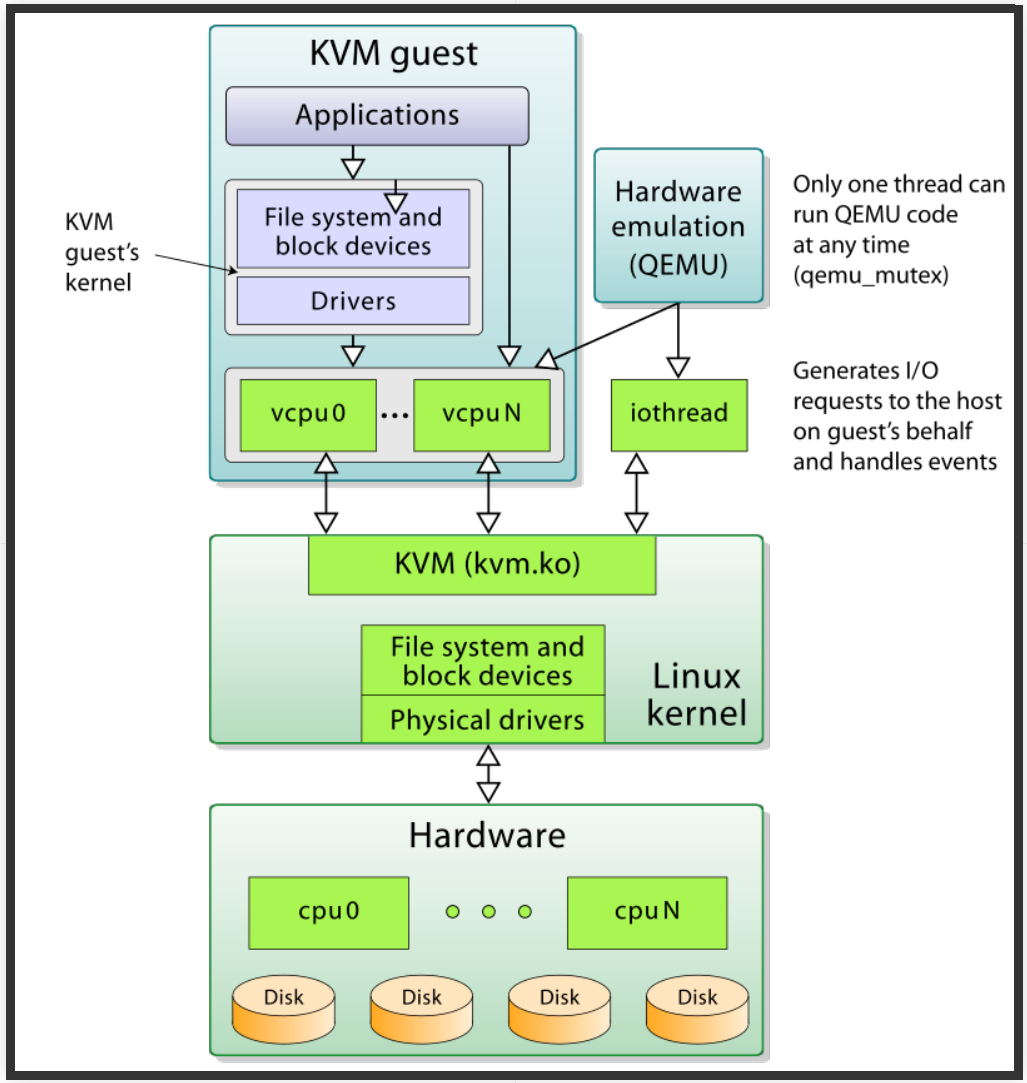
Проверка загрузочного образа https://www.system-rescue.org/:

```
qemu-system-x86_64 -boot d -cdrom systemrescuecd-amd64-6.1.4.iso
-m 1024
```

Загрузка с загрузочного образа с подключением образа диска виртуальной машины:

```
qemu-system-x86_64 -boot d -cdrom systemrescuecd-amd64-6.1.4.iso
-m 1024 -hda test-vm1.img
```

QEMU


KVM guest
Applications
File system and block devices
Drivers
KVM guest's kernel
vcpu 0
vcpu N
Hardware emulation (QEMU)
Only one thread can run QEMU code at any time (qemu_mutex)
iothread
Generates I/O requests to the host on guest's behalf and handles events
KVM (kvm.ko)
File system and block devices
Physical drivers
Linux kernel
Hardware
cpu 0
cpu N
Disk
Disk
Disk
Disk

Ваши вопросы?

Маршрут вебинара

- QEMU-KVM
- Установка KVM
- Управление KVM

Установка KVM

Установка KVM

Установка репозитория QEMU Enterprise Virtualization

```
yum install centos-release-qemu-ev
```

Установка QEMU и KVM

```
yum install qemu-kvm-ev
```

KVM nested virtualization

Проверяем поддержку вложенной виртуализации на процессоре хост-машины:

```
cat /sys/module/kvm_intel/parameters/nested
```

Выгружаем модуль ядра:

```
modprobe -r kvm_intel
```

Добавляем параметр к модулю ядра при его загрузке:

```
modprobe kvm_intel nested=1
```

KVM nested virtualization

Добавляем параметр в конфиг /etc/modprobe.d /kvm.conf:

```
echo "modprobe kvm_intel nested=1" > /etc/modprobe.d/kvm.conf
```

libvirt

libvirt

libvirt - это демон и API, с помощью которого пользовательские интерфейсы осуществляют управление гипервизорами

Поддерживаемые гипервизоры:

- LXC
- OpenVZ
- QEMU (KVM)
- Xen
- VirtualBox
- VMware ESXi, VMware Workstation, VMware Player
- Hyper-V

libvirt

libvirt - это демон и API, с помощью которого пользовательские интерфейсы осуществляют управление гипервизорами

Поддерживаемые пользовательские интерфейсы:

- virsh
- virt-manager
- oVirt
- mist.io

libvirt

libvirt - это демон и API, с помощью которого пользовательские интерфейсы осуществляют управление гипервизорами

Особенности:

- создание, редактирование, запуск и остановка виртуальных машин
- просмотр статистики и производительности
- управление как локальным, так и удаленными гипервизорами

Установка libvirt

Установка libvirt

Установка демона libvirtd и консольного клиента virsh:

```
yum install libvirt libvirt-client
```

Основной конфиг libvirtd - **/etc/libvirt/libvirtd.conf**

Ваши вопросы?

Маршрут вебинара

- QEMU-KVM
- Установка KVM
- Управление KVM

Интерфейс virsh

Интерфейс virsh

Обзор команд virsh:

https://docs.fedoraproject.org/ru-RU/Fedora/12/html/Virtualization_Guide/chap-Virtualization_Guide-Managing_guests_with_virsh.html

Интерфейс virsh

Основные команды virsh:

Список доменов (виртуальных машин):

```
virsh list
```

Запуск виртуальной машины:

```
virsh start vm-test1
```

Корректное завершение работы виртуальной машины:

```
virsh shutdown vm-test1
```

Интерфейс virsh

Poweroff виртуальной машины:

```
virsh destroy vm-test1
```

Удаление виртуальной машины:

```
virsh undefine vm-test1
```

Работа с удаленным гипервизором поверх подключения по ssh:

```
virsh -c qemu+ssh://user@kvm-host/system list
```

Работа с дисками

Работа с дисками

Создание диска в формате raw:

```
qemu-img create -f raw /var/lib/libvirt/images/test-vm1.img 40G
```

Создание диска в формате qcow2:

```
qemu-img create -f qcow2 /var/lib/libvirt/images/test-vm1.qcow 40
```

Конвертация диска из формата vhdx в формат raw:

```
qemu-img convert -p -f vhdx -O raw test-vm1.vhdx test-vm1.raw
```

Работа с дисками

Конвертация диска из формата qcow2 в формат vdi:

```
qemu-img convert -O vdi vda.qcow2 vda.vdi
```

Увеличение размера диска в формате qcow2:

```
qemu-img resize test-vm1.qcow2 +20G
```

Уменьшение размера образа диска в формате qcow2 путем копирования в новый образ:

```
qemu-img create -f qcow2 -o preallocation=metadata \
newimage.qcow2 NEW_SIZE
virt-resize oldimage.qcow2 newimage.qcow2
```

Snapshots

Snapshots

Создание снапшота (диск + память):

```
virsh snapshot-create test-vm1
```

Создание снапшота (только диск):

```
virsh snapshot-create test-vm1 --disk-only
```

Просмотр списка сделанных снапшотов:

```
virsh snapshot-list --domain logstash
```

Удаление снапшота:

```
virsh snapshot-delete test-vm1 1548863936
```

Создание виртуальной машины

Создание виртуальной машины

Пример создания виртуальной машины с помощью virt-install:

```
virt-install --connect qemu+ssh://user@kvm-host-ip/system \
--virt-type kvm --network bridge:virbr0 --name test-vm1 \
--os-variant centos7.0 --ram=1024 --vcpus=1 \
--disk size=4,bus=virtio --os-type=linux \
--graphics vnc,password=123456 \
--cdrom /var/lib/libvirt/boot/CentOS-7-x86_64-Minimal-2003.iso
```

Создание виртуальной машины

Пример создания виртуальной машины с диском LVM:

```
virt-install --connect qemu+ssh://user@kvm-host-ip/system \
--virt-type kvm --network bridge:virbr0 --name test-vm1 \
--os-variant centos7.0 --ram=1024 --vcpus=1 \
--disk path=/dev/mapper/vg0-test-vm1,bus=virtio --os-type=linux \
--graphics vnc,password=123456 \
--cdrom /var/lib/libvirt/boot/CentOS-7-x86_64-Minimal-2003.iso
```

Графическая консоль виртуальной машины

Пример использования virt-viewer:

```
virt-viewer --connect qemu+ssh://user@kvm-host-ip/system test-vm1
```

Редактирование конфигурации виртуальной машины

Редактирование конфигурации виртуальной машины

Подключаем диск с хост-машины в виртуальную машину

Редактируем конфиг виртуальной машины:

```
virsh edit test-vm1
```

```
...
<disk type='block' device='disk'>
<driver name='qemu' type='raw' cache='none'/>
<source dev='/dev/sdc'/>
<target dev='vda' bus='virtio'/>
</disk>
...
```

Редактирование конфигурации виртуальной машины

Подключаем usb устройство с хост-машины в виртуальную машину

Смотрим адреса устройств usb на хост-машине:

```
lsusb
...
Bus 003 Device 011: ID 0529:0620 Aladdin Knowledge Systems Token
Bus 003 Device 010: ID 0529:0620 Aladdin Knowledge Systems Token
...
```

Редактирование конфигурации виртуальной машины

Создаем конфиги для каждого устройства:

```
vim usb1.xml
<hostdev mode='subsystem' type='usb'>
  <source>
    <address bus='3' device='11'/>
  </source>
</hostdev>
```

```
vim usb2.xml
<hostdev mode='subsystem' type='usb'>
  <source>
    <address bus='3' device='11'/>
  </source>
</hostdev>
```

Редактирование конфигурации виртуальной машины

Добавляем конфиги в основной конфиг виртуальной машины:

```
virsh attach-device test-vm1 usb1.xml
virsh attach-device test-vm1 usb2.xml
```

Редактирование конфигурации виртуальной машины

Используем возможности технологии NUMA:

```
<cpu>
    <numa>
        <cell id="0" cpus="0-1" memory="3" unit="GiB"/>
        <cell id="1" cpus="2-3" memory="3" unit="GiB"/>
    </numa>
</cpu>
```

Импорт/экспорт конфигурации виртуальной машины

Импорт/экспорт конфигурации виртуальной машины

Экспорт конфигурации виртуальной машины:

```
virsh dumpxml test-vm1 > test-vm1.xml
```

Импорт конфигурации виртуальной машины:

```
virsh define /home/user/test-vm1.xml
```

Импорт/экспорт конфигурации виртуальной машины

Важный момент. На гипервизорах разных версий на разных дистрибутивах могут не совпадать:

- пути к исполняемым файлам qemu-kvm
- типы виртуальных машин
- модели процессоров
- пути к файлами образов дисков виртуальных машин

Ваши вопросы?

Миграция виртуальной машины

Миграция виртуальной машины

Основные условия:

- образы виртуальных машин на обоих гипервизорах должны лежать по абсолютно одинаковому пути
- для удобства лучше, если пользователь, от имени которого будет происходить перенос, может беспрепятственно заходить на соседний гипервизор по ssh (с помощью ключей)
- чем быстрее сеть между гипервизорами - тем быстрее произойдет миграция

Миграция виртуальной машины

Создаем диск виртуальной машины на сервере назначения:

```
qemu-img create -f qcow2 -o preallocation=metadata \
/var/lib/libvirt/images/test-vm1.qcow2 4.1G
```

Запускаем миграцию на исходном сервере:

```
virsh migrate --live --copy-storage-all test-vm1 \
qemu+ssh://kvm2-server-ip/system \
--undefinesource --persistent
```

virt-manager

virt-manager

virt-manager - приложение от RedHat, позволяющее управлять виртуальными машинами

virt-manager

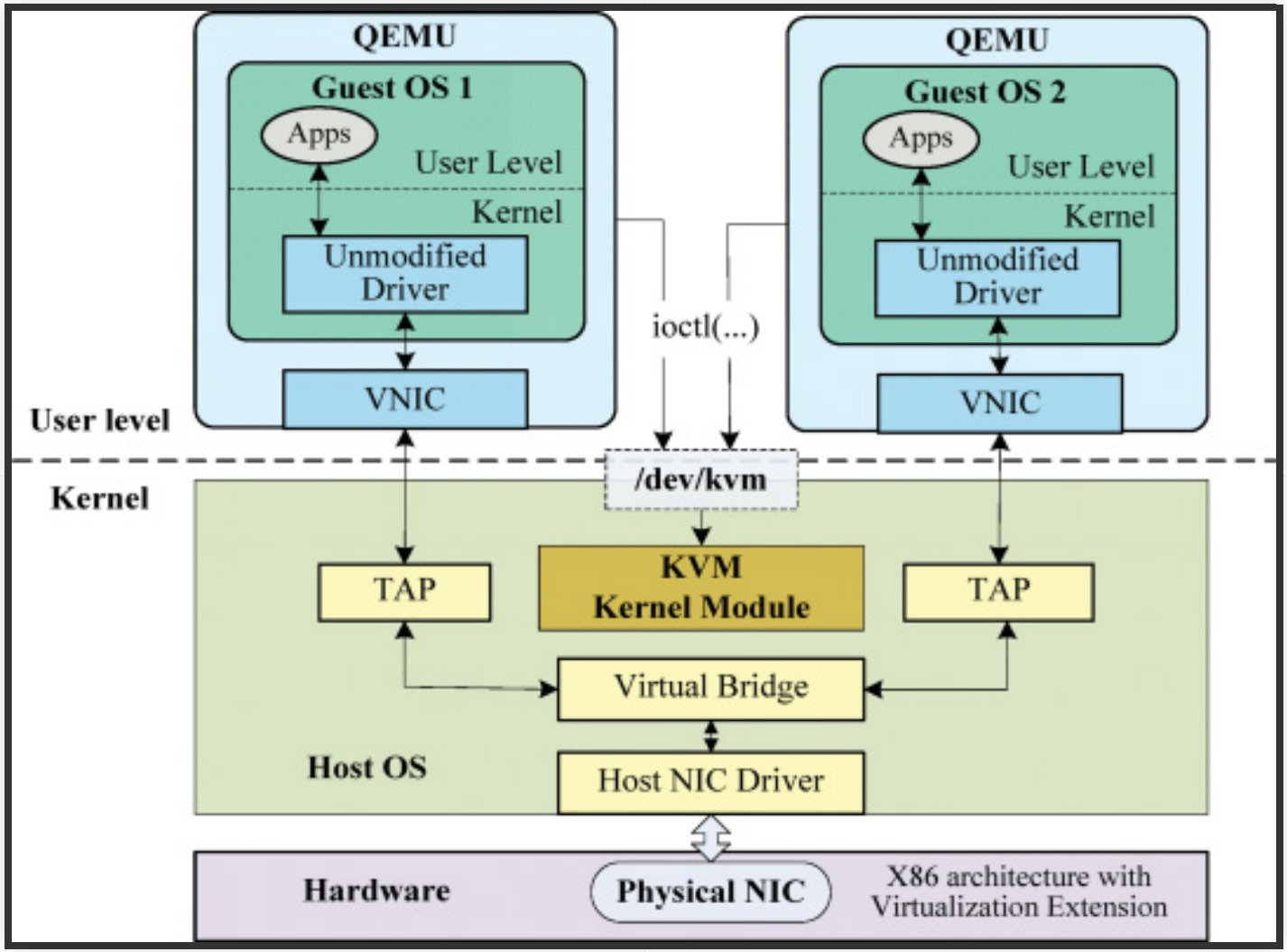
Особенности:

- создание, редактирование, запуск и остановку виртуальных машин
- просмотр и управление каждой виртуальной машиной посредством консоли
- просмотр производительности и статистики по каждой виртуальной машине
- просмотр всех запущенных виртуальных машин на хосте и их производительность
- можно использовать KVM, Xen или QEMU виртуальные машины, запущенные локально или удаленно

Ваши вопросы?

Сеть в KVM

Сеть в KVM


QEMU
Guest OS 1
Apps
User Level
Kernel
Unmodified Driver
VNIC
QEMU
Guest OS 2
Apps
User Level
Kernel
Unmodified Driver
VNIC
ioctl(...)
User level
Kernel
/dev/kvm
TAP
KVM Kernel Module
TAP
Virtual Bridge
Host OS
Host NIC Driver
Hardware
Physical NIC
X86 architecture with Virtualization Extension

Сеть в KVM

Просмотр сетей гипервизора:

```
virsh net-list
```

Просмотр краткой информации по сети default (сеть по-умолчанию):

```
virsh net-info default
```

Просмотр полной конфигурации сети default:

```
virsh net-dumpxml default
```

Просмотр аренды dhcp-сервера:

```
virsh net-dhcp-leases default
```

Сеть в KVM

Импорт конфигурации новой сети:

```
virsh net-define new_network.xml
```

Запуск новой сети:

```
virsh net-start br1-network
```

Сеть в KVM

Правила iptables для типовой сети

Таблица NAT, цепочка POSTROUTING:

```
iptables -A POSTROUTING -s 10.16.64.0/24 \
-d 224.0.0.0/24 -j RETURN
iptables -A POSTROUTING -s 10.16.64.0/24 \
-d 255.255.255.255/32 -j RETURN
iptables -A POSTROUTING -s 10.16.64.0/24 ! \
-d 10.16.64.0/24 -p tcp -j MASQUERADE --to-ports 1024-65535
iptables -A POSTROUTING -s 10.16.64.0/24 ! \
-d 10.16.64.0/24 -p udp -j MASQUERADE --to-ports 1024-65535
iptables -A POSTROUTING -s 10.16.64.0/24 ! \
-d 10.16.64.0/24 -j MASQUERADE
```

Сеть в KVM

Цепочка INPUT:

```
iptables -A INPUT -i br1 -p udp -m udp --dport 53 -j ACCEPT
iptables -A INPUT -i br1 -p tcp -m tcp --dport 53 -j ACCEPT
iptables -A INPUT -i br1 -p udp -m udp --dport 67 -j ACCEPT
iptables -A INPUT -i br1 -p tcp -m tcp --dport 67 -j ACCEPT
```

Сеть в KVM

Цепочки FORWARD и OUTPUT:

```
iptables -A FORWARD -d 10.16.64.0/24 -o br1 -j ACCEPT
iptables -A FORWARD -s 10.16.64.0/24 -i br1 -j ACCEPT
iptables -A FORWARD -d 10.16.64.0/24 -o br1 -m conntrack \
--ctstate RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i br1 -o br1 -j ACCEPT
iptables -A FORWARD -o br1 -j REJECT \
--reject-with icmp-port-unreachable
iptables -A FORWARD -i br1 -j REJECT \
--reject-with icmp-port-unreachable
iptables -A OUTPUT -o br1 -p udp -m udp --dport 68 -j ACCEPT
```

Ваши вопросы?

Заполните, пожалуйста, опрос о занятии по ссылке в чате

Приходите на следующие вебинары

Спасибо за внимание!