УТВЕРЖДЕН

ЛКНВ.11100-01 92 01-ЛУ

ОПЕРАЦИОННАЯ СИСТЕМА АЛЬТ 8 СП

(ОС Альт 8 СП)

Руководство администратора.

Виртуализация

ЛКНВ.11100-01 92 01

Листов 84

2020

Литера О

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

АННОТАЦИЯ

Настоящий документ содержит сведения о средствах виртуализации программного изделия (ПИ) «Операционная система Альт 8 СП» ЛКНВ.11100-01, сокращенное наименование – ОС Альт 8 СП, для варианта исполнения **Сервер** на архитектурах **Intel x86_64**, **AArch64 (ARMv8)**, **ppc64le (POWER)**.

Версия документа **1.0**.

Документ предназначен для администратора ОС Альт 8 СП и содержит общие сведения о структуре, настройке и работе с системой виртуализации ОС Альт 8 СП и компонентами виртуальной инфраструктуры.

СОДЕРЖАНИЕ

1. Общие сведения 5
2. LXC 8
3. Управление виртуализацией на основе libvirt 9
3.1. Установка и настройка libvirt 9
3.2. Утилиты управления 10
3.2.1. Утилита Virsh 11
3.2.2. Утилита virt-install 13
3.2.3. Утилита qemu-img 16
3.2.4. Менеджер ВМ virt-manager 17
3.3. Подключение к гипервизору 18
3.3.1. Управление доступом к libvirt через SSH 18
3.3.2. Подключение к сессии гипервизора с помощью virsh 19
3.3.3. Настройка соединения с удаленным гипервизором в virt-manager 20
3.4. Создание ВМ 21
3.4.1. Создание ВМ на основе файла конфигурации 21
3.4.2. Создание ВМ с помощью virt-install 22
3.4.3. Создание ВМ с помощью virt-manager 32
3.5. Управление ВМ 36
3.5.1. Управление конфигурацией ВМ 36
3.5.2. Управление виртуальными сетевыми интерфейсами и сетями 43
3.5.3. Управление хранилищами 48
3.6. Запуск и управление функционированием ВМ 53
3.6.1. Управление состоянием ВМ в командной строке 53
3.6.2. Управление состоянием ВМ в менеджере ВМ 54
3.7. Миграция ВМ 56
3.7.1. Миграция с помощью virsh 56
3.7.2. Миграция с помощью virt-manager 57
3.8. Снимки машины 59

3.8.1. Управления снимками ВМ в консоли 59
3.8.2. Управления снимками ВМ virt-manager 60
3.9. Управление доступом в виртуальной инфраструктуре 62
3.9.1. Пример тонкой настройки 67
3.10. Регистрация событий 69
3.10.1. Регистрация событий libvirt 69
3.10.2. Регистрация событий запуска (завершения) работы компонентов виртуальной инфраструктуры 71
3.10.3. Регистрация входа (выхода) субъектов доступа в/из гипервизор(а) ... 72
3.10.4. Регистрация событий входа (выхода) субъектов доступа в/из гостевых ОС 72
3.10.5. Регистрация изменения прав доступа к файлам-образам ВМ 72
4. Docker 73
4.1. Запуск от пользователя 73
5. Kubernetes 74
5.1. Подготовка 74
5.2. Разворачивание кластера 74
5.3. Тестовый запуск nginx 76
6. Удаленное подключение к ВМ 78
6.1. VNC подключение к ВМ 78
6.1.1. Подключение VNC-клиента к удаленному компьютеру 79
6.1.2. Отключение VNC-клиента от удаленного компьютера 80
6.2. SPICE подключение к ВМ 80
6.3. Проброс USB-устройств в ВМ через SPICE 82
Перечень сокращений 83

1. ОБЩИЕ СВЕДЕНИЯ

ОС Альт 8 СП Сервер обладает следующими функциональными характеристиками:

- обеспечивает возможность обработки, хранения и передачи информации;
- обеспечивает возможность функционирования в многозадачном режиме (одновременное выполнение множества процессов);
- обеспечивает возможность масштабирования системы: возможна эксплуатация операционной системы (ОС) как на одной персональной электронно-вычислительной машине (ПЭВМ), так и в информационных системах различной архитектуры;
- обеспечивает многопользовательский режим эксплуатации;
- обеспечивает поддержку мультипроцессорных систем;
- обеспечивает поддержку виртуальной памяти;
- обеспечивает поддержку запуска виртуальных машин (ВМ);
- обеспечивает сетевую обработку данных, в том числе разграничение доступа к сетевым пакетам.

В ОС Альт 8 СП выполняются следующие функциональные требования безопасности к виртуальной инфраструктуре:

- идентификация и аутентификация субъектов доступа и объектов доступа в виртуальной инфраструктуре, в том числе администраторов управления средствами виртуализации;
- управление доступом субъектов доступа к объектам доступа в виртуальной инфраструктуре, в том числе внутри ВМ;
- регистрация событий безопасности в виртуальной инфраструктуре;
- управление (фильтрация, маршрутизация, контроль соединения, однонаправленная передача) потоками информации между компонентами виртуальной инфраструктуры, а также по периметру виртуальной инфраструктуры;

- управление перемещением ВМ (контейнеров) и обрабатываемых на них данных;
- контроль целостности виртуальной инфраструктуры и ее конфигураций;
- резервное копирование данных, резервирование технических средств, программного обеспечения (ПО) виртуальной инфраструктуры, а также каналов связи внутри виртуальной инфраструктуры;
- разбиение виртуальной инфраструктуры на сегменты (сегментирование виртуальной инфраструктуры) для обработки информации отдельным пользователем и (или) группой пользователей.

Дистрибутив ОС Альт 8 СП предоставляет администратору возможность размещать системные службы (сервисы) в изолированных окружениях, ВМ.

Управление системой виртуализации возможно через командный интерфейс и веб-интерфейс, с использованием API.

ОС Альт 8 СП поддерживает клиент-серверную архитектуру и может обслуживать процессы как в пределах одной компьютерной системы, так и процессы на других ПЭВМ через каналы передачи данных или сетевые соединения.

ОС Альт 8 СП Сервер предоставляет средства виртуализации и набор дополнительных служб, востребованных в инфраструктуре виртуализации любой сложности и архитектуры:

1) базовый гипервизор (libvirt, qemu-kvm);
2) контейнерная виртуализация (docker, kubernetes, podman, lxc);
3) ПО для организации хранилища:
   - распределенная сетевая файловая система CEPH;
   - распределенная сетевая файловая система GlusterFS;
   - сервер сетевой файловой системы NFS;
   - поддержка iSCSI как в качестве клиента, так и создание сервера;
4) ПО для сети:
   - сетевые службы DNS и DHCP;
   - виртуальный сетевой коммутатор Open vSwitch;

- служба динамической маршрутизации bird с поддержкой протоколов BGP, OSPF и др.;
- сетевой балансировщик нагрузки HAProxy, keepallived;
- веб-серверы Apache и Nginx;

5) ПО для мониторинга (zabbix-agent, prometheus-node_exporter);
6) ПО резервного копирования (Bacula).

2. LXC

Система виртуализации LXC (Linux Containers) работает на уровне ОС для запуска нескольких изолированных экземпляров ОС семейства Linux на одном узле.

LXC не использует ВМ, а создает виртуальное окружение с собственным пространством процессов и сетевым стеком. Все экземпляры LXC используют один экземпляр ядра ОС.

По применению доступно множество материалов в сети Интернет, рекомендуется, например, ресурс http://xgu.ru/wiki/LXC.

3. УПРАВЛЕНИЕ ВИРТУАЛИЗАЦИЕЙ НА ОСНОВЕ LIBVIRT

3.1. Установка и настройка libvirt

Виртуализацию как таковую можно по подходу разбить на несколько типов:

- полная виртуализация;
- паравиртуализация;
- виртуализация окружения.

Различные типы виртуализации реализуются различными системами виртуализации. В ОС Альт 8 СП не поддерживается паравиртуализация. Полная виртуализация представлена системой виртуализации KVM (Kernel-based Virtual Machine), а виртуализация окружения представлена системой LXC (LinuxContainers).

Различные системы виртуализации представляют различные интерфейсы для управления ВМ и контейнерами. Однако набор действий, производимых с ВМ и контейнерами, не меняется от системы виртуализации к системе виртуализации, меняется лишь способ указания системе виртуализации произвести те или иные действия. Эта особенность позволила создать некоторый общий API и набор утилит для управления ВМ, поддерживающий различные системы виртуализации.

libvirt – это набор инструментов, предоставляющий единый API к множеству различных технологий виртуализации.

Кроме управления ВМ/контейнерами libvirt поддерживает управление виртуальными сетями и управление хранением образов.

Для управления из консоли разработан набор утилит `virt-install`, `virt-clone`, `virsh` и других.

Для управления из графической оболочки, например, на компьютере с ОС Альт 8 СП Рабочая станция, можно воспользоваться virt-manager.

Любой виртуальный ресурс, необходимый для создания ВМ (`compute`, `network`, `storage`) представлен в виде объекта в libvirt. За процесс описания и создания этих объектов отвечает набор различных XML-файлов. Сама ВМ в терминологии libvirt называется доменом (`domain`). Это тоже объект внутри libvirt, который описывается отдельным XML-файлом.

При первоначальной установке и запуске libvirt по умолчанию создает мост (`bridge`) `virbr0` и его минимальную конфигурацию. Этот мост не будет подключен ни к одному физическому интерфейсу, однако, может быть использован для связи ВМ внутри одного гипервизора.

Для развертывания libvirt в уже установленной системе, достаточно установить пакеты:

```
# apt-get update
# apt-get install libvirt-kvm virt-install
```

Запуск службы:

```
# systemctl start libvirtd
# systemctl enable libvirtd
```

Для непривилегированного доступа (не `root`) к управлению libvirt, нужно добавить пользователя в группу `vmusers`:

```
# usermod -a -G vmusers user
```

Сервер виртуализации использует следующие каталоги хостовой файловой системы:

- `/etc/libvirt/` – каталог с файлами конфигурации libvirt;
- `/var/lib/libvirt/` – рабочий каталог сервера виртуализации libvirt;
- `/var/log/libvirt` – файлы журналов libvirt.

3.2. Утилиты управления

Основные утилиты командной строки для управления ВМ:

- qemu-img – управление образами дисков ВМ. Позволяет выполнять операции по созданию образов различных форматов, конвертировать файлы-образы между этими форматами, получать информацию об образах и объединять снимки ВМ для тех форматов, которые это поддерживают;

- virsh – консольный интерфейс управления ВМ, виртуальными дисками и виртуальными сетями;
- virt-clone – клонирование ВМ;
- virt-convert – конвертирования ВМ между различными форматами и программно-аппаратными платформами;
- virt-image – создание ВМ по их XML описанию;
- virt-install – создание ВМ с помощью опций командной строки;
- virt-xml – редактирование XML-файлов описаний ВМ.

3.2.1. Утилита Virsh

virsh – утилита для командной строки, предназначенная для управления ВМ и гипервизорами KVM.

virsh использует libvirt API и служит альтернативой графическому менеджеру ВМ (virt-manager).

С помощью virsh можно сохранять состояние ВМ, переносить ВМ между гипервизорами и управлять виртуальными сетями.

В таблице 1 и таблице 2 приведены основные параметры утилиты командной строки virsh.

Для получения списка доступных команд или параметров, выполните команду:

```
$ virsh help
```

Т а б л и ц а 1 – Команды управления ВМ

Команда	Описание
`help`	Краткая справка
`list`	Просмотр всех ВМ
`dumpxml`	Вывести файл конфигурации XML для заданной ВМ
`create`	Создать ВМ из файла конфигурации XML и ее запуск
`start`	Запустить неактивную ВМ
`destroy`	Принудительно остановить работу ВМ

Окончание таблицы 1

Команда	Описание
`define`	Определяет файл конфигурации XML для заданной ВМ
`domid`	Просмотр идентификатора ВМ
`domuuid`	Просмотр UUID ВМ
`dominfo`	Просмотр сведений о ВМ
`domname`	Просмотр имени ВМ
`domstate`	Просмотр состояния ВМ
`quit`	Закрыть интерактивный терминал
`reboot`	Перезагрузить ВМ
`restore`	Восстановить сохраненную в файле ВМ
`resume`	Возобновить работу приостановленной ВМ
`save`	Сохранить состояние ВМ в файл
`shutdown`	Корректно завершить работу ВМ
`suspend`	Приостановить работу ВМ
`undefine`	Удалить все файлы ВМ
`migrate`	Перенести ВМ на другой узел

Т а б л и ц а 2 – Параметры управления ресурсами ВМ и гипервизора

Команда	Описание
`setmem`	Определяет размер выделенной ВМ памяти
`setmaxmem`	Ограничивает максимально доступный гипервизору объем памяти
`setvcpus`	Изменяет число предоставленных ВМ виртуальных процессоров
`vcpuinfo`	Просмотр информации о виртуальных процессорах
`vcpupin`	Настройка соответствий виртуальных процессоров
`domblkstat`	Просмотр статистики блочных устройств для работающей ВМ
`domifstat`	Просмотр статистики сетевых интерфейсов для работающей ВМ
`attach-device`	Подключить определенное в XML-файле устройство к ВМ

Окончание таблицы 2

Команда	Описание
`attach-disk`	Подключить новое дисковое устройство к ВМ
`attach-interface`	Подключить новый сетевой интерфейс к ВМ
`detach-device`	Отключить устройство от ВМ (принимает те же определения XML, что и `attach-device`)
`detach-disk`	Отключить дисковое устройство от ВМ
`detach-interface`	Отключить сетевой интерфейс от ВМ

3.2.2. Утилита virt-install

virt-install – это инструмент для создания ВМ, основанный на командной строке.

Далее подробно рассматриваются возможности создания ВМ при помощи утилиты командной строки virt-install. В таблице 3 приведено описание только наиболее часто используемые опции virt-install. Описание всех доступных опций можно получить, выполнив команду:

```
$ man virt-install
```

Утилита virt-install поддерживает как графическую установку ОС при помощи VNC и Spice, так и текстовую установку через последовательный порт. Гостевая система может быть настроена на использование нескольких дисков, сетевых интерфейсов, аудиоустройств и физических USB- и PCI-устройств.

Установочный носитель может располагаться как локально, так и удаленно, например, на NFS, HTTP или FTP серверах. В последнем случае virt-install получает минимальный набор файлов для запуска установки и позволяет установщику получить отдельные файлы. Также поддерживается загрузка по сети (PXE) и создание ВМ/контейнера без установки ОС.

Утилита virt-install поддерживает большое число опций, позволяющих создать полностью независимую ВМ, готовую к работе, что хорошо подходит для автоматизации установки ВМ.

Т а б л и ц а 3 – Параметры команд virt-install

Опции	Описание
`-n NAME`, `--name=NAME`	Имя новой ВМ. Это имя должно быть уникально внутри одного гипервизора.
`--memory MEMORY`	Определяет размер выделенной ВМ памяти (в Мбайт).

`--vcpus VCPUS`	Определяет количество виртуальных центральных процессорных устройств (ЦПУ). Например:
	`--vcpus 5`
	`--vcpus 5,maxvcpus=10,cpuset=1-4,6,8`
	`--vcpus sockets=2,cores=4,threads=2`

`--cpu CPU`	Модель ЦП и его характеристики. Например:
	`--cpu coreduo,+x2apic`
	`--cpu host-passthrough`
	`--cpu host`

`--metadata METADATA`	Метаданные ВМ.
Метод установки	
`--cdrom CDROM`	Установочный CD-ROM. Может указывать на файл ISO-образа или на устройство чтения CD/DVD-дисков.
`-l LOCATION`, `--location LOCATION`	Источник установки, например, `https://host/path`.
`--pxe`	Выполнить загрузку из сети используя протокол PXE.
`--import`	Пропустить установку ОС, и создать ВМ на основе существующего образа диска.
`--boot BOOT`	Параметры загрузки ВМ. Например: `--boot hd,cdrom,menu=on` `--boot init=/sbin/init` (для контейнеров)
`--os-type=DISTRO_TYPE`	Оптимизирует настройки ВМ для заданного типа ОС.
`--os-variant=DISTRO_VARIANT`	Дополнительная оптимизация ВМ для конкретного варианта ОС.

`--disk DISK`	Настройка пространства хранения данных. Например:
	`--disk size=10` (новый образ на 10 Гбайт в выбранном по умолчанию месте)
	`--disk /my/existing/disk,cache=none`
	`--disk device=cdrom,bus=scsi`
	`--disk=?`

Окончание таблицы 3

Опции	Описание

`-w NETWORK,`
`--network NETWORK`
	Конфигурация сетевого интерфейса ВМ. Например:
	`--network bridge=mybr0`
	`--network network=my_libvirt_virtual_net`
	`--network network=mynet,model=virtio,mac=00:11...`
	`--network none`

`--graphics GRAPHICS`	Настройки экрана ВМ. Например:
	`--graphics spice`
	`--graphics vnc,port=5901,listen=0.0.0.0`
	`--graphics none`

`--input INPUT`	Конфигурация устройства ввода. Например:
	`--input tablet`
	`--input keyboard,bus=usb`

`--hostdev HOSTDEV`	Конфигурация физических USB/PCI и других устройств хоста для совместного использования ВМ.
`--filesystem FILESYSTEM`	Передача каталога хоста гостевой системе. Например: `--filesystem /my/source/dir,/dir/in/guest`
Параметры платформы виртуализации	
`-v,` `--hvm`	Эта ВМ должна быть полностью виртуализированной.
`-p,` `--paravirt`	Эта ВМ должна быть паравиртуализированной.
`--container`	Тип ВМ – контейнер.
`--virt-type VIRT_TYPE`	Тип гипервизора (kvm, qemu и т. п.).
`--arch ARCH`	Имитируемая архитектура процессора.
`--machine MACHINE`	Имитируемый тип компьютера.
Прочие параметры	
`--autostart`	Запускать домен автоматически при запуске хоста.
`--transient`	Создать временный домен.
`--noautoconsole`	Не подключаться к гостевой консоли автоматически.
`-q,` `--quiet`	Подавлять вывод (за исключением ошибок).
`-d,` `--debug`	Вывести отладочные данные.

3.2.3. Утилита qemu-img

qemu-img – инструмент для манипулирования образами дисков машин QEMU.

Использование:

```
qemu-img command [command options]
```

Для манипуляции с образами используются следующие команды:

- `create` – создание нового образа диска;
- `check` – проверка образа диска на ошибки;
- `convert` – конвертация существующего образа диска в другой формат;
- `info` – получение информации о существующем образе диска;
- `snapshot` – управляет снимками состояний (`snapshot`) существующих образов дисков;
- `commit` – записывает произведенные изменения на существующий образ диска;
- `rebase` – создает новый базовый образ на основании существующего.

qemu-img работает со следующими форматами:

- `raw` – простой формат для дисковых образов, обладающий отличной переносимостью на большинство технологий виртуализации и эмуляции. Только непосредственно записанные секторы будут занимать место на диске. Действительный объем пространства, занимаемый образом, можно определить с помощью команд `qemu-img info` или `ls -ls`;
- `qcow2` – формат QEMU. Этот формат рекомендуется использовать для небольших образов (в частности, если файловая система не поддерживает фрагментацию), дополнительного шифрования AES, сжатия zlib и поддержки множества снимков ВМ;
- `qcow` – старый формат QEMU. Используется только в целях обеспечения совместимости со старыми версиями;
- `cow` – формат COW (Copy On Write). Используется только в целях обеспечения совместимости со старыми версиями;
- `vmdk` – формат образов, совместимый с VMware 3 и 4;

- `cloop` – формат CLOOP (Compressed Loop). Его единственное применение состоит в обеспечении повторного использования сжатых напрямую образов CD-ROM, например, Knoppix CD-ROM.

Команда получения сведений о дисковом образе:

```
# qemu-img info /var/lib/libvirt/images/alt8.0.qcow2
image: /var/lib/libvirt/images/alt8.0.qcow2
file format: qcow2
virtual size: 12 GiB (12884901888 bytes)
disk size: 12 GiB
cluster_size: 65536
Format specific information:
     compat: 1.1
     lazy refcounts: true
     refcount bits: 16
     corrupt: false
```

В результате будут показаны сведения о запрошенном образе, в том числе зарезервированный объем на диске, а также информация о снимках ВМ.

Команда создания образа для жесткого диска (динамически расширяемый):

```
# qemu-img create –f qcow2 /var/lib/libvirt/images/hdd.qcow2 20G
```

Команда конвертирования образ диска из формата `raw` в `qcow2`:

```
# qemu-img convert -f raw -O qcow2 disk_hd.img disk_hd.qcow2
```

3.2.4. Менеджер ВМ virt-manager

Менеджер ВМ virt-manager предоставляет графический интерфейс для доступа к гипервизорам и ВМ в локальной и удаленных системах. С помощью virt-manager можно создавать ВМ. Кроме того, virt-manager выполняет управляющие функции:

- выделение памяти;
- выделение виртуальных процессоров;
- мониторинг производительности;
- сохранение и восстановление, приостановка и возобновление работы, запуск и завершение работы ВМ;
- доступ к текстовой и графической консоли;
- автономная и живая миграция.

Для запуска менеджера ВМ, в меню приложений необходимо выбрать «Система» → «Менеджер виртуальных машин» («Manage virtual machines»).

Примечание. Должен быть установлен пакет virt-manager.

В главном окне менеджера (рис. 1) показаны все запущенные ВМ и выделенные им ресурсы. Поля можно отфильтровать. Двойной щелчок на имени ВМ открывает ее консоль. Выбор ВМ и двойной щелчок на кнопке «Подробности» («Details») откроет окно сведений об этой машине.

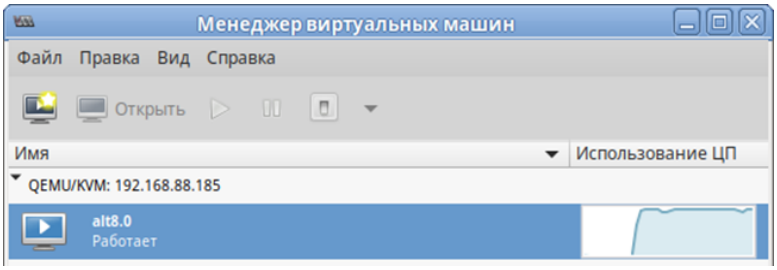

Рис. 1 – Главное окно менеджера ВМ

3.3. Подключение к гипервизору

3.3.1. Управление доступом к libvirt через SSH

В дополнение к аутентификации SSH также необходимо определить управление доступом для службы libvirt в хост-системе (рис. 2).

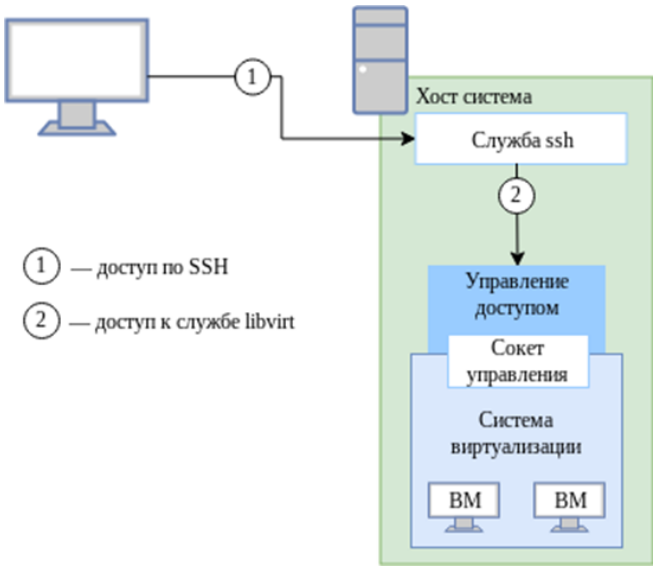

Рис. 2 – Доступ к libvirt с удаленного узла

Для настройки подключения к удаленному серверу виртуализации на узле, с которого будет производиться подключение, необходимо сгенерировать SSH-ключ и скопировать его публичную часть на сервер. Для этого с правами пользователя, от имени которого будет создаваться подключение, требуется выполнить в консоли следующие команды:

```
$ ssh-keygen -t rsa
$ ssh-copy-id user@192.168.88.248
```

где `192.168.88.248` – IP-адрес сервера с libvirt.

В результате появится возможность работы с домашними каталогами пользователя user на сервере с libvirt.

Для доступа к libvirt достаточно добавить пользователя `user` в группу `vmusers` на сервере, либо скопировать публичный ключ пользователю `root` и подключаться к серверу по ssh от имени `root` – `root@server`.

3.3.2. Подключение к сессии гипервизора с помощью virsh

Команда подключения к гипервизору:

```
virsh -c URI
```

Если параметр URI не задан, то libvirt попытается определить наиболее подходящий гипервизор.

Параметр URI может принимать следующие значения:

- `qemu:///system` – подключиться к службе, которая управляет KVM/QEMU-доменами и запущена под `root`. Этот вариант используется по умолчанию для пользователей virt-manager;
- `qemu:///session` – подключиться к службе, которая управляет KVM/QEMU-доменами и запущена от имени непривилегированного пользователя;
- `lxc:///` – подключиться к гипервизору для создания LXC контейнеров (должен быть установлен пакет libvirt-lxc).

Чтобы установить соединение только для чтения, к приведенной выше команде следует добавить опцию `--readonly`.

Пример создания локального подключения:

```
$ virsh -c qemu:///system list --all
 ID          Имя                Статус
----------------------------------------------------
 -          alt8.0             выключен
```

Подключение к удаленному гипервизору QEMU через протокол SSH:

```
$ virsh -c qemu+ssh://user@192.168.88.185/system

Добро пожаловать в virsh – интерактивный терминал виртуализации.
Введите «help» для получения справки по командам «quit», чтобы завершить работу и выйти.
virsh #
```

где:

- `user` – имя пользователя на удаленном хосте, который входит в группу `vmusers`;
- `192.168.88.185` – IP-адрес или имя хоста ВМ.

3.3.3. Настройка соединения с удаленным гипервизором в virt-manager

virt-manager позволяет управлять несколькими удаленными хостами ВМ.

Для создания нового подключения необходимо в меню менеджера ВМ выбрать «Файл» → «Добавить соединение…».

В открывшемся окне необходимо выбрать сессию гипервизора, отметить пункт «Connect to remote host over SSH» и ввести имя пользователя и адрес сервера (рис. 3).

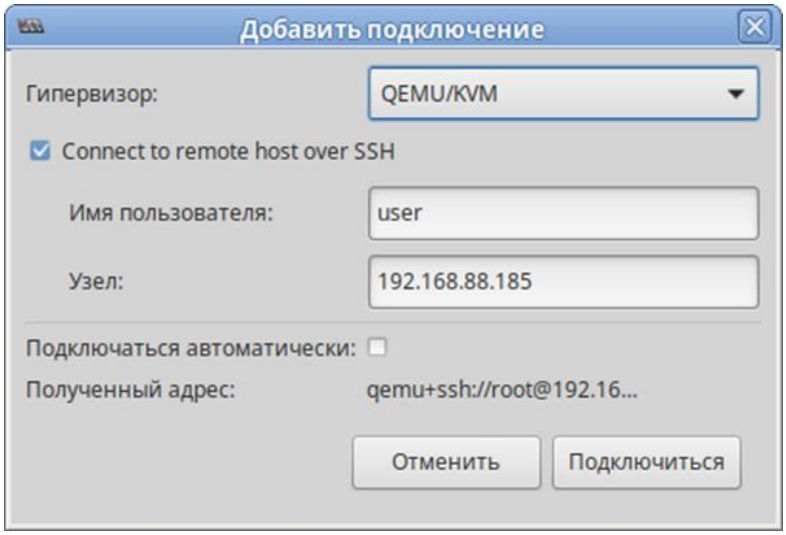

Рис. 3 – Окно соединений менеджера ВМ

Примечание. На управляющей системе можно запустить virt-manager, выполнив следующую команду:

```
virt-manager -c qemu+ssh://user@192.168.88.185/system
```

где:

- `user` – имя пользователя на удаленном хосте, который входит в группу `vmusers`;
- `192.168.88.185` – IP-адрес или имя хоста ВМ.

3.4. Создание ВМ

Наиболее важным этапом в процессе использования виртуализации является создание ВМ. Именно при создании ВМ задается используемый тип виртуализации, способы доступа к ВМ, подключение к локальной сети и другие характеристики виртуального оборудования.

Установка ВМ может быть запущена из командной строки с помощью программы virt-install или из пользовательского интерфейса программы virt-manager.

3.4.1. Создание ВМ на основе файла конфигурации

ВМ могут быть созданы из файлов конфигурации. Можно сделать копию существующего XML-файла ранее созданной ВМ, или использовать опцию `dumpxml`.

Вывод файла конфигурации ВМ:

```
# virsh dumpxml <domain-id, domain-name или domain-uuid>
```

Эта команда выводит XML-файл конфигурации ВМ в стандартный вывод (`stdout`). Можно сохранить данные, отправив вывод в файл.

Пример передачи вывода в файл `guest.xml`:

```
# virsh dumpxml alt8.0 > guest.xml
```

Можно отредактировать этот файл конфигурации, чтобы настроить дополнительные устройства или развернуть дополнительные ВМ.

Команда создания ВМ из XML файла:

```
# virsh create guest.xml
```

3.4.2. Создание ВМ с помощью virt-install

Утилита virt-install поддерживает как графическую установку ОС при помощи VNC и Spice, так и текстовую установку через последовательный порт. Гостевая система может быть настроена на использование нескольких дисков, сетевых интерфейсов, аудиоустройств и физических USB- и PCI-устройств.

Установочный носитель может располагаться как локально, так и удаленно, например, на NFS-, HTTP- или FTP-серверах. В последнем случае `virt-install` получает минимальный набор файлов для запуска установки и позволяет установщику получить отдельные файлы. Также поддерживается загрузка по сети (PXE) и создание ВМ/контейнера без установки ОС.

Утилита `virt-install` поддерживает большое число опции, позволяющих создать полностью независимую ВМ, готовую к работе, что хорошо подходит для автоматизации установки ВМ.

Минимальные требуемые опции: `--name`, `--ram`, хранилище (`--disk`, `--filesystem` или `--nodisks`) и опции установки. Далее описаны только наиболее часто используемые опции.

Чтобы использовать команду `virt-install`, необходимо сначала загрузить ISO-образ той ОС, которая будет устанавливаться.

Команда создания ВМ:

```
# virt-install --connect qemu:///system
--name alt-server \
--os-type=linux \
--os-variant=alt.p9 \
--cdrom /var/lib/libvirt/images/alt-server-v-x86_64.iso \
--graphics vnc\
--disk pool=default,size=20,bus=virtio,format=qcow2 \
--ram 2048 \
--vcpus=2 \
--network network=default \
--hvm \
--virt-type=kvm
```

где:

- `--name alt-server` – название ВМ;
- `--os-type=linux` – тип ОС;
- `--os-variant=alt.p9` – версия ОС;
- `--cdrom /var/lib/libvirt/images/alt-server-v-x86_64.iso` – путь к ISO-образу установочного диска ОС;
- `--graphics vnc` – графическая консоль;
- `--disk pool=default, size=20, bus=virtio, format=qcow2` – хранилище. ВМ будет создана в пространстве хранения объемом 20 Гбайт, которое автоматически выделяется из пула хранилищ `default`. Образ диска для этой виртуальной машины будет создан в формате `qcow2`;
- `--ram 2048` – объем оперативной памяти;
- `--vcpus=2` – количество процессоров;
- `--network network=default` – виртуальная сеть `default`;
- `--hvm` – полностью виртуализированная система;
- `--virt-type=kvm` – использовать модуль ядра KVM, который задействует аппаратные возможности виртуализации процессора.

Последние две опции команды `virt-install` оптимизируют ВМ для использования в качестве полностью виртуализированной системы (`--hvm`) и указывают, что KVM является базовым гипервизором (`--virt-type`) для поддержки новой ВМ. Обе этих опции обеспечивают определенную оптимизацию в процессе создания и установки ОС; если эти опции не заданы в явном виде, то выше указанные значения применяются по умолчанию.

Можно использовать подобную команду для создания ВМ, исполняющей другую ОС. С этой целью нужно задать надлежащее имя для ВМ и соответствующим образом изменить аргументы опций `--cdrom`, `--os-type` и `--os-variant`.

Список доступных вариантов ОС можно получить, выполнив команду:

```
$ osinfo-query os
```

3.4.2.1. Общие опции

Общие опции утилиты virt-install:

1) `-h`, `--help` – показать помощь и выйти;
2) `-q`, `--quiet` – печатать только сообщения о критических ошибках;
3) `-d`, `--debug` – выводить отладочную информацию в терминал. Отладочная информация выводится в файл `$HOME/.virtinst/virt-install.log` даже если эта опция не указана;
4) `--connect=URI` – подключить к заданному гипервизору. Если данная опция не задана, то libvirt попытается определить наиболее подходящий гипервизор. Доступные значения:
   - `qemu:///system` – для создания KVM и QEMU гостей, запускаемых системным экземпляром libvirtd. Этот вариант используется по умолчанию для пользователей virt-manager;
   - `qemu:///session` – для созданий KVM и QEMU гостей, запускаемых от имени обычного пользователя;
   - `xen:///` – для создания Xen;
   - `lxc:///` – для создания LXC контейнеров;
5) `-n ИМЯ`, `--name=ИМЯ` – задает имя новой ВМ. Это имя должно быть уникально внутри одного гипервизора. Для того чтобы переопределить существующего гостя следует сначала воспользоваться virsh для остановки и удаления его (`virsh shutdown` и `virsh undefine`);
6) `-r ПАМЯТЬ`, `--ram=ПАМЯТЬ` – задает объем оперативной памяти гостя в Мбайтах. Если у гипервизора недостаточно памяти для того чтобы назначить ее гостю, то он автоматически возьмет недостающую часть у хост-системы;
7) `--arch=архитектура` – задает «не родную» архитектуру процессора для ВМ. Если данная опция не указана, то будет использована та же архитектура процессора что и у процессора хост-системы;

8) `--vcpus=ВИРТCPU[,maxvcpus=МАКСИМУМ][,sockets=#][,cores=#][,threads=#]` – задает число виртуальных процессоров для гостя. Если задано значение для `maxvcpus`, то гость будет иметь возможность подключать до `МАКСИМУМ` виртуальных процессоров, но запускаться он будет с `ВИРТCPU`. Также для виртуального процессора можно задать число сокетов, ядер и нитей. Если какие-то из этих значений не указаны, то они будут автоматически вычислены.

3.4.2.2. Опции метода установки

Опции метода установки утилиты `virt-install`:

1) `-c CDROM`, `--cdrom=CDROM` – для гостей с полной виртуализацией задает файл или устройство, которое будет использоваться как устройство CD-ROM. Может указываться на файл ISO-образа или на устройство чтения CD/DVD-дисков. Также может быть URL до ISO-образа. Формат URL такой же как и в опции `--location`;
2) `-l РАСПОЛОЖЕНИЕ`, `--location=РАСПОЛОЖЕНИЕ` – указывает расположение дистрибутива для установки. Для некоторых дистрибутивов virt-install может распознать ядро и initrd, и получить их перед запуском установки.

   `РАСПОЛОЖЕНИЕ` может указываться в следующих формах:

   - `ДИРЕКТОРИЯ` – путь до локальной директории, содержащей установочный образ дистрибутива;
   - `nfs:хост:/путь` или `nfs://хост/путь` – NFS-путь, по которому доступен установочный образ дистрибутива;
   - `http://хост/путь` – HTTP-путь, по которому доступен установочный образ дистрибутива;
   - `ftp://хост/путь` – FTP-путь, по которому доступен установочный образ дистрибутива;
3) `--import` – пропустить установку ОС, и создать ВМ с существующим диском. В качестве загрузочного устройства будет использоваться первое указанной с опцией `--disk` или `--filesystem`;

4) `--init=ПУТЬ_К_INIT` – задает путь для бинарного файла, который будет использоваться в гостевой системе в качестве процесса init. Если корневая файловая система задана через `--filesystem`, то по умолчанию будет использоваться `/sbin/init`, в противном случае – `/bin/sh`;
5) `--livecd` – указывает, что установочный диск является LiveCD и что следует настроить ВМ на постоянную загрузку с CDROM. Может быть полезно в сочетании с опцией `--nodisks`;
6) `-x ДОПОЛНИТЕЛЬНО`, `--extra-args=ДОПОЛНИТЕЛЬНО` – дополнительные аргументы ядра, передаваемые в процессе установки (если указана опция `location`);
7) `--os-type=ТИП_ОС` – оптимизирует настройки ВМ для типа ВМ. Будет произведена попытка выбрать наиболее подходящие настройки для ACPI, APIC, подобрать оптимальные драйвера для «мыши» и virtio, а также произвести другие специфичные для ОС настройки ВМ. По умолчанию virt-install пытается автоматически определить ТИП_ОС на основании установочного носителя. Автоопределение можно отключить, используя значение none;
8) `--os-variant=ВАРИАНТ_ОС` – дополнительные оптимизации ВМ для конкретного варианта ОС. Данная опция не является обязательной и не требует указания опции `--os-type`. По умолчанию virt-install пытается автоматически определить `ВАРИАНТ_ОС` на основании установочного носителя. Автоопределение можно отключить, используя значение none. Если задано значение `list`, то virt-install напечатает список доступный вариантов ОС.

3.4.2.3. Опции хранилища

Опции хранилища утилиты virt-install:

`--disk=ОПЦИИ_ДИСКА` – задает носитель для использования в качестве хранилища в ВМ. Общий формат следующий:

```
--disk опция1=значение1,опция2=значение2,...
```

Для задания носителя может использоваться сокращенный формат:

`--disk /some/storage/path,опция1=значение1`

Или же может быть использован один из следующих аргументов:

- `path` – путь к какому-либо носителю (существующему или не существующему). Существующий носитель может быть либо файлом, либо блочным устройством. При установке на удаленной хост-системе существующий носитель должен быть общим как том хранилища libvirt. Указание несуществующего пути подразумевает попытку создать новое хранилище и требует задание значения `size`. Если директория в `path` это пул хранилища libvirt на хост-системе, то новое хранилище будет создано как том хранилища libvirt. Для удаленных систем директория должна указывать на пул хранилища;
- `pool` – имя существующего пула хранилищ, в котором будет создано новое хранилище. Также требует указания значения `size`;
- `vol` – имя существующего тома в хранилище libvirt. Указывается как `poolname/volname`;
- `size` – размер в Гбайт для создаваемых хранилищ;
- `--nodisks` – указывает, что у ВМ не должно быть логических дисков для хранения информации. Обычно данная опция используется для запуска LiveCD образом или при установке на сетевое хранилище типа iSCSI или NFS.

3.4.2.4. Опции сети

Опции сети утилиты virt-install:

`-w СЕТЬ, --network=СЕТЬ,опция1=значение1,опция2=значение2` – подключить ВМ к сети.

Значение СЕТЬ может задаваться в трех формах:

- `bridge=МОСТ` – подключить к устройству типа мост с именем `МОСТ` в хост-системе. Используйте эту опцию, если в хост-системе заданы постоянные сетевые настройки и гостевая система требует прямого

взаимодействия с локальной сетью. Также `bridge` следует использовать, если требуется живая миграция ВМ;

- `network=ИМЯ` – подключить к виртуальной сети хост-системы под названием `ИМЯ`. Виртуальные сети можно просматривать, создавать и удалять при помощи утилиты командной строки virsh. По умолчанию при установке libvirt создается сеть с именем `default`. Используйте виртуальную сеть в случае, если в хост-системе могут меняться сетевые настройки (например, при помощи NetworkManager) или же используется WiFi. Для пакетов из ВМ будет применяться трансляция адресов в подключенную в данный момент локальную сеть;
- `user` – подключить к локальной сети при помощи SLiRP. Используйте только при запуске ВМ QEMU от непривилегированного пользователя. Предоставляет очень ограниченный вариант трансляции адресов.

Если данная опция не указана, то будет создан один сетевой интерфейс. Если на физическом интерфейсе в хост-системе создан мостовой интерфейс, то будет использован именно он. В противном случае будет использовано подключение к виртуальной сети `default`. Эту опцию можно указать несколько раз для создания нескольких интерфейсов.

Другие доступные опции:

- `model` – задать сетевое устройство (как оно будет отображаться в ВМ). В качестве значений следует использовать поддерживаемые гипервизором, например, `e1000`, `rtl8139`, `virtio` и другие;
- `mac` – задать MAC-адрес для ВМ. Если данный параметр не указан (или указано значение `RANDOM`), то будет сгенерирован случайный MAC-адрес. Для ВМ QEMU и KVM MAC-адрес должен начинаться с 52:54:00;
- `--nonetworks` – используется для создания ВМ без сетевых интерфейсов.

3.4.2.5. Опции графики

Если не заданы графические опции, то, в зависимости от того задана переменная окружения `DISPLAY` или нет, будет использоваться по умолчанию опция `–graphicsvnc` или `--graphicsnone` соответственно.

`--graphics ТИП,опция1=аргумент1,опция2=аргумент2,...` – задает настройки для виртуального монитора. Данная опция не влияет на оборудование ВМ, она лишь задает способ доступа к монитору ВМ.

Поддерживаются следующие опции:

1) `type` – тип дисплея. Может быть одним из:
   - `vnc` – предоставить доступ к дисплею ВМ при помощи VNC (при этом используется адрес хост-системы). Если не задать параметр `port`, то будет использован первый свободный порт выше `5900`. Актуальное значение параметров VNC, назначенные ВМ, можно получить при помощи команды `virsh vncdisplay`;
   - `sdl` – открыть дисплей ВМ в SDL-окне на хост-системе. Если это окно закрыть, то ВМ будет также закрыта;
   - `spice` – экспортировать дисплей ВМ по протоколу Spice. Spice поддерживает возможности передачи аудио и USB-устройств, а также имеет высокую графическую производительность;
   - `none` – графическая консоль не будет подключена в ВМ. Для ВМ с полной виртуализацией (Xen и QEMU/KVM) потребуется на первом последовательном порту гостевой системы (этого можно добиться при помощи опции `--extra-args`). Для подключения к последовательному устройству можно использовать `virsh console ИМЯ`;
2) `port` – задать постоянный порт для дисплея ВМ. Используется совместно с `vnc` и `spice`.

3.4.2.6. Опции виртуализации

Опции виртуализации утилиты virt-install:

- `-v`, `--hvm` – если на хост-системе доступны полная виртуализация и паравиртуализация, то будет использована полная виртуализация. Эта опция недоступна при подключении к XEN без аппаратной поддержки виртуализации. Эта опция предполагается по умолчанию при подключении к гипервизору QEMU;

- `--container` – гостевая машина должна быть типа контейнер. Данную опцию следует использовать, только если хост-система также поддерживает другие типы виртуализации. Она пол предполагается по умолчанию для LXC и OpevVZ, но может быть явно указана для полноты;
- `--virt-type` – задает тип используемого гипервизора, например, kvm, qemu, xen или kqemu. Доступные типы виртуализации можно увидеть в тегах `domain`.

3.4.2.7. Опции устройств

Опции устройств утилиты virt-install:

- `--serial=ОПЦИИ` – задает подключение последовательного устройства к ВМ.

Общий формат следующий:

- `--serialtype,опция1=значение1,опция2=значение2,...`
- `--serialpty` – псевдо-TTY. Назначенное pty-устройство можно будет узнать из XML-описания ВМ;
- `--serialfile,path=ИМЯ_ФАЙЛА` – записывать вывод в файл ИМЯ_ФАЙЛА.

3.4.2.8. Примеры установки ОС в гостевые системы

Установка Fedora 13 в гостевую систему на базе KVM с диском и сетью работающими по virtio, с созданием файла хранилища размеров 8 Гбайт, с установкой с CD/DVD диска, находящегося в приводе хост-системы, а также с автоматическим запуском VNC-клиента:

```
# virt-install \
 --connect qemu:///system \
 --virt-type kvm \
 --name demo \
 --ram 500 \
 --disk path=/var/lib/libvirt/images/demo.img,size=8 \
 --graphics vnc \
 --cdrom /dev/cdrom \
 --os-variant fedora13
```

Установка Fedora 9 в гостевую систему на базе QEMU, с LVM разделом, подключением к виртуальной сети, загрузкой по сети и использование VNC для доступа к дисплею:

```
# virt-install \
 --connect qemu:///system \
 --name demo \
 --ram 500 \
 --disk path=/dev/HostVG/DemoVM \
 --network network=default \
 --virt-type qemu
 --graphics vnc \
 --os-variant fedora9
```

Установка FedoraCore 6 в гостевую систему на базе QEMU с архитектурой процессора, отличной от архитектуры хост-системы, использованием SDL для доступа к дисплею ВМ, а также с использованием удаленных ядра и initrd:

```
# virt-install \
 --connect qemu:///system \
 --name demo \
 --ram 500 \
 --disk path=/dev/hdc \
 --network bridge=eth1 \
 --arch ppc64 \
 --graphics sdl \
 --location
http://download.fedora.redhat.com/pub/fedora/linux/core/6/x86_64/
os/
```

Запуск Live CD в ВМ без дисков:

```
# virt-install \
 --hvm \
 --name demo \
 --ram 500 \
 --nodisks \
 --livecd \
 --graphics vnc \
 --cdrom /var/lib/libvirt/images/altlive.iso
```

Запуск `/usr/bin/httpd` в контейнере (LXC), с ограничением памяти в 512 Мбайт и двумя ядрами хост-системы:

```
# virt-install \
 --connect lxc:/// \
 --name httpd_guest \
 --ram 512 \
 --vcpus 2 \
 --init /usr/bin/httpd
```

Создать ВМ, используя существующий том хранилища:

```
# virt-install \
 --name demo \
 --ram 512 \
 --disk /home/user/VMs/mydisk.img \
 --import
```

Тестировать отдельное ядро и initrd при запуске существующего тома хранилища с привязкой последовательного порта ВМ к `ttyS0` на хост-системе:

```
# virt-install \
 --name mykernel \
 --ram 512 \
 --disk /home/user/VMs/mydisk.img \
--boot
kernel=/tmp/mykernel,initrd=/tmp/myinitrd,kernel_args="console=tt
yS0" \
 --serial pty
```

3.4.3. Создание ВМ с помощью virt-manager

Создание новой ВМ:

- нажать кнопку «Создать виртуальную машину» в главном окне virt-manager, либо
- выбрать в меню «Файл» → «Создать виртуальную машину».

На первом шаге создания ВМ необходимо выбрать метод установки ОС (рис. 4) и нажать на кнопку «Вперед».

В следующем окне для установки гостевой ОС требуется указать ISO-образ установочного диска ОС или CD/DVD-диск с дистрибутивом (рис. 5).

Данное окно будет выглядеть по-разному в зависимости от выбора, сделанного на предыдущем этапе.

Здесь также можно указать версию устанавливаемой ОС.

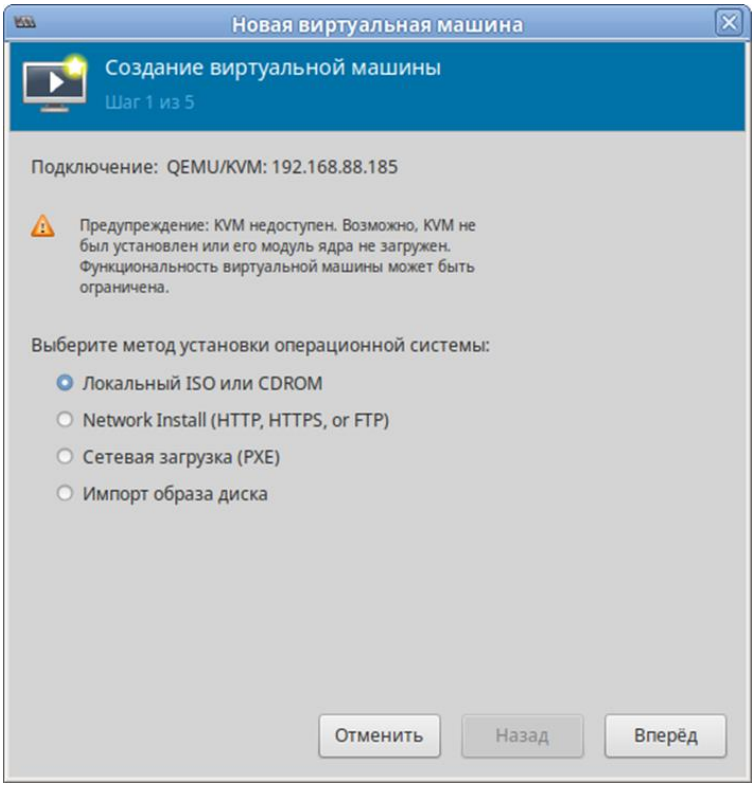

Рис. 4 – Создание ВМ. Выбор метода установки

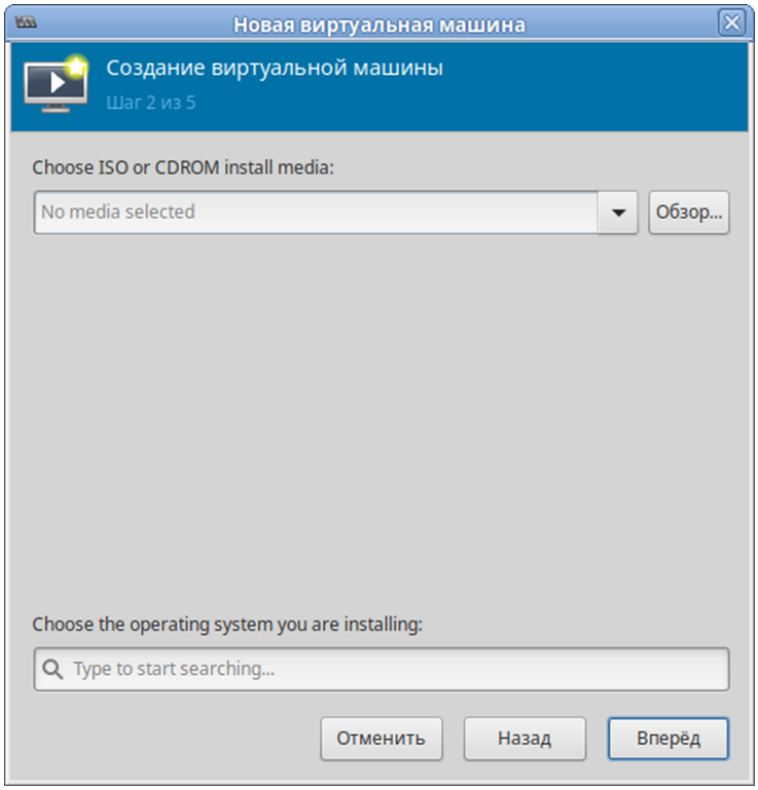

Рис. 5 – Создание ВМ. Выбор ISO образа

На третьем шаге необходимо указать размер памяти и количество процессоров для ВМ (рис. 6). Эти значения влияют на производительность хоста и ВМ.

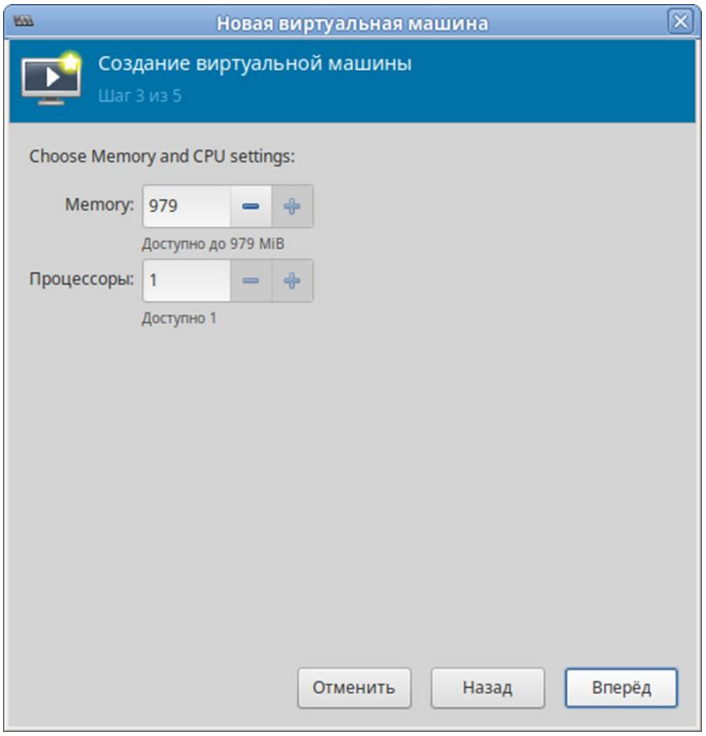

Рис. 6 – Создание ВМ. Настройка оперативного запоминающего устройства (ОЗУ) и ЦПУ для ВМ

На следующем этапе настраивается пространство хранения данных (рис. 7).

На последнем этапе (рис. 8) можно задать название ВМ, выбрать сеть и нажать на кнопку «Готово».

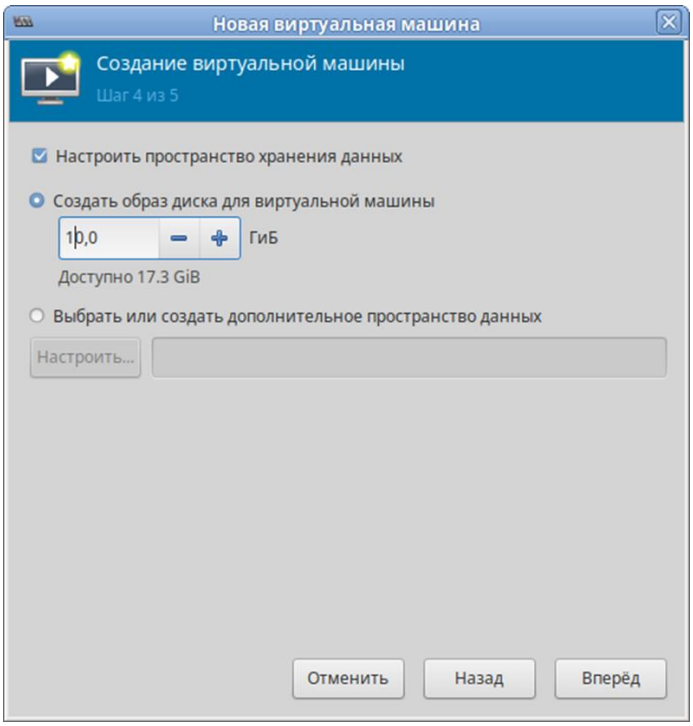

Рис. 7 – Создание ВМ. Настройка пространства хранения данных

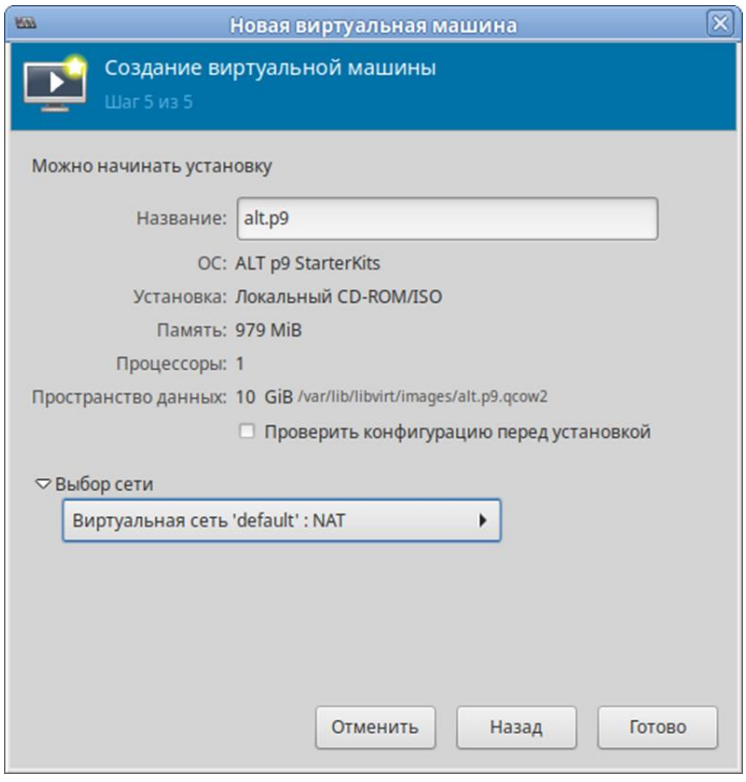

Рис. 8 – Создание ВМ. Выбор сети

В результате созданная ВМ будет запущена и после завершения исходной загрузки начнется стандартный процесс установки ОС.

Окружение локального рабочего стола способно перехватывать комбинации клавиш (например, <Ctrl>+<Alt>+<F11>) для предотвращения их отправки гостевой машине. Чтобы отправить такие последовательности, используется свойство «западания» клавиш virt-manager. Для перевода клавиши в нажатое состояние необходимо нажать клавишу модификатора (<Ctrl> или <Alt>) 3 раза. Клавиша будет считаться нажатой до тех пор, пока не будет нажата любая клавиша, отличная от модификатора. Таким образом, чтобы передать гостевой системе комбинацию клавиш <Ctrl>+<Alt>+<F11>, необходимо последовательно нажать клавиши <Ctrl>+<Ctrl>+<Ctrl>+<Alt>+<F11> или воспользоваться меню «Отправить комбинацию клавиш».

3.5. Управление ВМ

3.5.1. Управление конфигурацией ВМ

3.5.1.1. Редактирование файла конфигурации ВМ

ВМ могут редактироваться либо во время работы, либо в автономном режиме. Эту функциональность предоставляет команда `virsh edit`. Например, команда редактирования ВМ с именем `alt-server`:

```
# virsh edit alt-server
```

В результате выполнения этой команды откроется окно текстового редактора, заданного переменной оболочки `$EDITOR`.

3.5.1.2. Получение информации о ВМ

Команда для получения информации о ВМ:

```
virsh dominfo <domain-id, domain-name or domain-uuid>
```

Пример вывода `virsh dominfo`:

```
$ virsh dominfo alt8.0
ID:       -
Имя:      alt8.0
UUID:           e645d4c4-4044-42cc-af17-91ef146dcd9d
Тип ОС:         hvm
Статус:         выключен
CPU:            1
Макс.память: 512000 KiB
Занято памяти: 512000 KiB
```

```
Постоянство: yes
Автозапуск: выкл.
Управляемое сохранение: no
Модель безопасности: none
DOI безопасности: 0
```

Команда получения информации об узле:

```
virsh nodeinfo
```

Пример вывода `virsh nodeinfo`:

```
$ virsh nodeinfo
Модель процессора: x86_64
CPU:                   1
Частота процессора: 1995 MHz
Сокеты:             1
Ядер на сокет:      1
Потоков на ядро: 1
Ячейки NUMA:        1
Объем памяти: 1003296 KiB
```

Вывод содержит информацию об узле и машинах, поддерживающих виртуализацию.

Просмотр списка ВМ:

```
virsh list
```

Опции команды `virsh list`:

- `--inactive` – показать список неактивных доменов;
- `--all` – показать все ВМ независимо от их состояния.

Пример вывода `virsh list`:

```
$ virsh list --all
ID   Имя            Статус

----------------------------------------
8    alt-server     работает
```

Столбец «Статус» может содержать следующие значения:

- работает (running) – работающие ВМ, то есть те машины, которые используют ресурсы процессора в момент выполнения команды;
- blocked – заблокированные, неработающие машины. Такой статус может быть вызван ожиданием ввода/вывода или пребыванием машины в спящем режиме;

- приостановлен (paused) – приостановленные домены. В это состояние они переходят, если администратор нажал кнопку паузы в окне менеджера ВМ или выполнил команду `virsh suspend`. В приостановленном состоянии ВМ продолжает потреблять ресурсы, но не может занимать больше процессорных ресурсов;
- выключен (shutdown) – ВМ, завершающие свою работу. При получении ВМ сигнала завершения работы, она начнет завершать все процессы (некоторые ОС не отвечают на такие сигналы);
- dying – сбойные домены и домены, которые не смогли корректно завершить свою работу;
- crashed – сбойные домены, работа которых была прервана. В этом состоянии домены находятся, если не была настроена их перезагрузка в случае сбоя.

Команда получения информации о виртуальных процессорах:

```
virsh vcpuinfo <domain-id, domain-name or domain-uuid>
```

Пример вывода:

```
# virsh vcpuinfo alt-server
VCPU:     0
CPU:      0
Статус:   работает
Время CPU: 115,3s
Соответствие CPU: y
```

Команда сопоставления виртуальных процессоров физическим:

```
virsh vcpupin <domain-id, domain-name or domain-uuid> vcpu, cpulist
```

Здесь `vcpu` – номер виртуального процессора, а `cpulist` – сопоставляемые ему физические процессоры.

Команда изменения числа процессоров для домена (заданное число не может превышать значение, определенное при создании ВМ):

```
virsh setvcpus <domain-id, domain-name or domain-uuid> count [-- maximum] [--config] [--live] [--current] [--guest]
```

где:

- [--count] <число> – число виртуальных процессоров;
- --config – с сохранением после перезагрузки;
- --live – применить к работающему домену;
- --current – применить к текущему домену;
- --guest – состояние процессоров ограничивается гостевым доменом.

Команда изменения выделенного ВМ объема памяти:

```
virsh setmem <domain-id or domain-name> size [--config] [--live] [-- current]
```

где:

- [--size] <число> – целое значение нового размера памяти (по умолчанию в Кбайт);
- --config – с сохранением после перезагрузки;
- --live – применить к работающему домену;
- --current – применить к текущему домену.

Объем памяти, определяемый заданным числом, должен быть указан в Кбайт. Объем не может превышать значение, определенное при создании ВМ, но в то же время не должен быть меньше 64 Мбайт. Изменение максимального объема памяти может оказать влияние на функциональность ВМ только в том случае, если указанный размер меньше исходного. В таком случае использование памяти будет ограничено.

Команда изменения максимальное ограничение памяти:

```
virsh setmaxmem <domain-id or domain-name> size [--config] [--live] [--current]
```

где:

- [--size] <число> – целое значение максимально допустимого размера памяти (по умолчанию в Кбайт);
- --config – с сохранением после перезагрузки;
- --live – применить к работающему домену;
- --current – применить к текущему домену.

Примеры изменения размера оперативной памяти и количества виртуальных процессоров соответственно:

```
# virsh --connect qemu:///system setmaxmem --size 624000 alt8.0
# virsh --connect qemu:///system setmem --size 52240 alt8.0
# virsh --connect qemu:///system setvcpus --config alt8.0 3 --maximum
```

Команда для получения информации о блочных устройствах работающей ВМ:

```
virsh domblkstat GuestName <block-device>
```

Команда для получения информации о сетевых интерфейсах работающей ВМ:

```
virsh domifstat GuestName <interface-device>
```

3.5.1.3. Конфигурирование ВМ в менеджере ВМ

С помощью менеджера ВМ можно получить доступ к подробной информации обо всех ВМ, для этого следует:

1) в главном окне менеджера выбрать ВМ;
2) нажать на кнопку «Открыть» (рис. 9);
3) в открывшемся окне нажать на кнопку «Показать виртуальное оборудование» (рис. 10);
4) появится окно просмотра сведений ВМ.

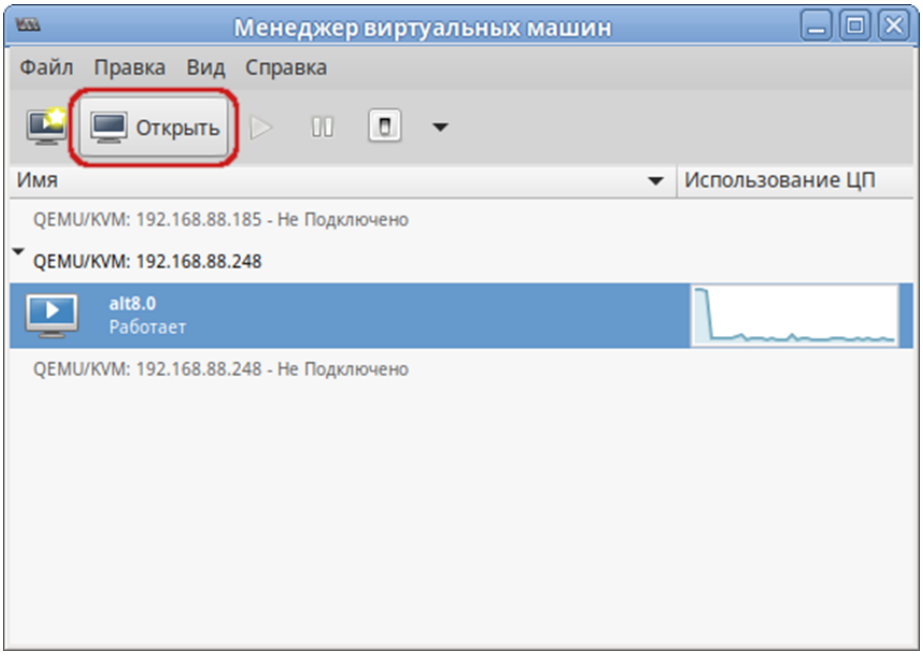

Рис. 9 – Окно менеджера ВМ

Для изменения требуемого параметра необходимо перейти на нужную вкладку (например, рис. 11, рис. 12), внести изменения и подтвердить операцию – нажать на кнопку «Применить».

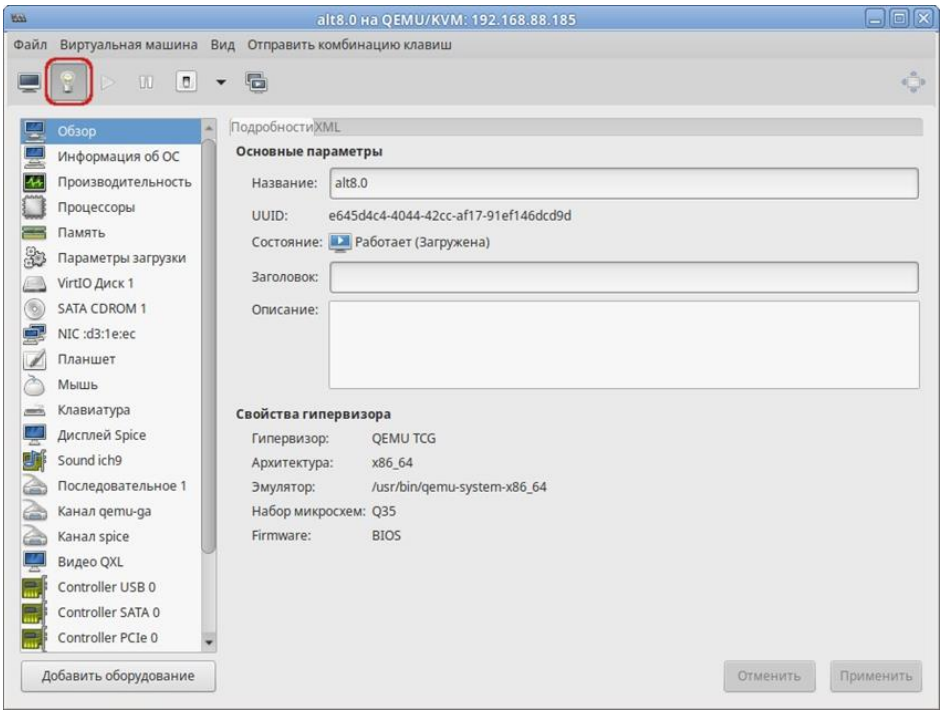

Рис. 10 – Окно параметров ВМ

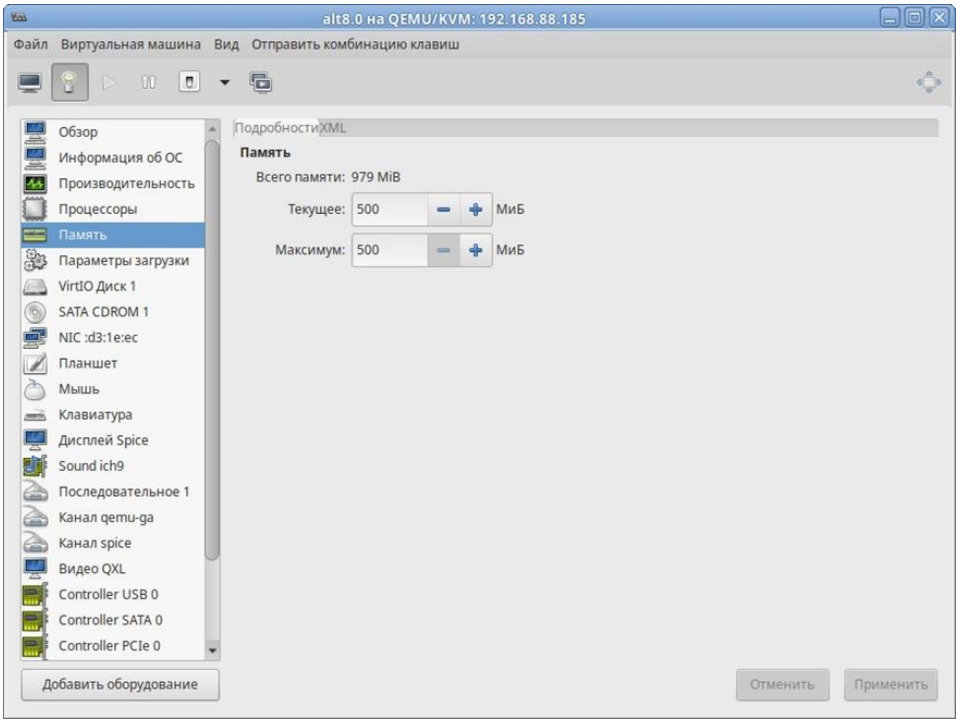

Рис. 11 – Вкладка «Память»

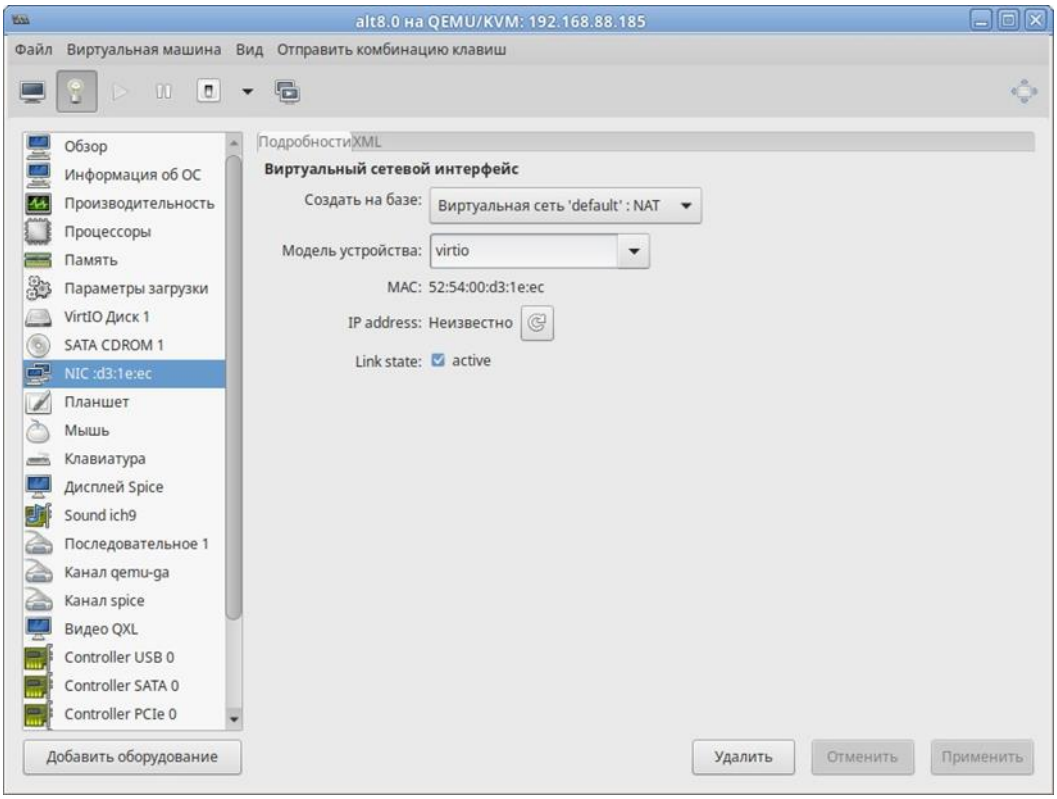

Рис. 12 – Вкладка «Сеть»

3.5.1.4. Мониторинг состояния

С помощью менеджера ВМ можно изменить настройки контроля состояния ВМ.

Для этого в меню «Правка» следует выбрать пункт «Параметры», в открывшемся окне «Настройки» на вкладке «Статистика» можно задать время обновления состояния ВМ в секундах (рис. 13).

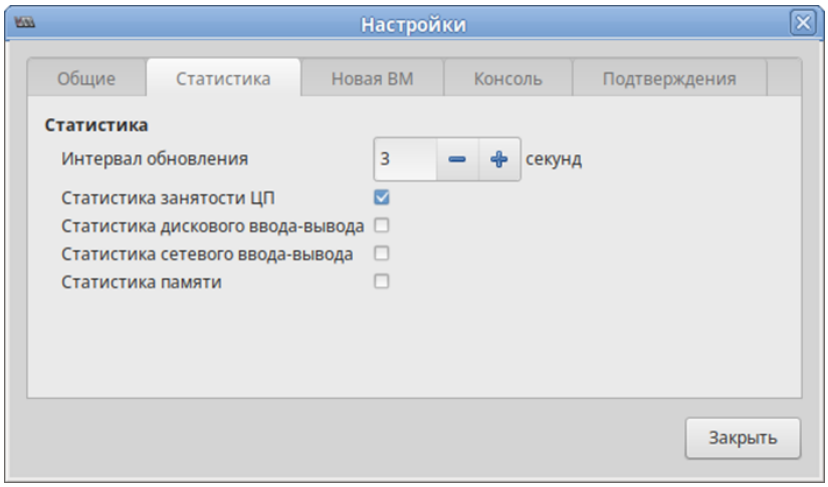

Рис. 13 – Вкладка «Статистика»

Во вкладке «Консоль» (рис. 14) можно выбрать, как открывать консоль, и указать устройство ввода.

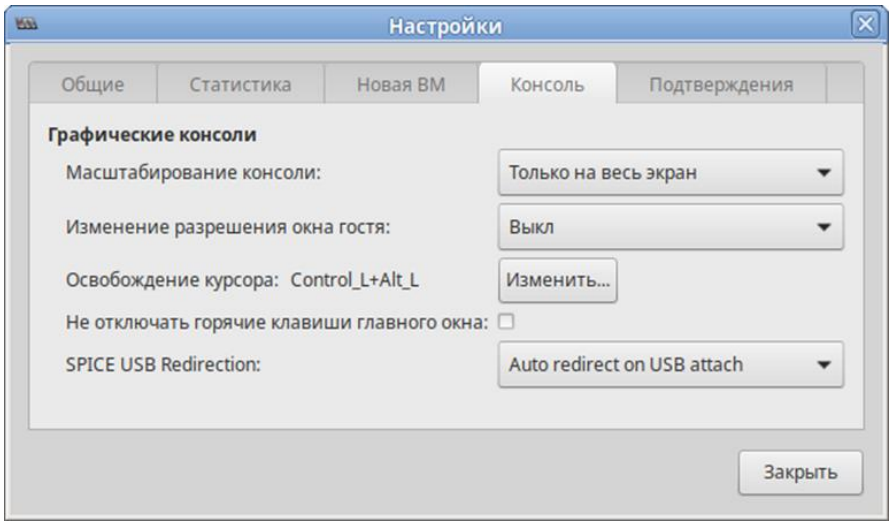

Рис. 14 – Вкладка «Консоль»

3.5.2. Управление виртуальными сетевыми интерфейсами и сетями

При базовых настройках используется виртуальная сеть недоступная извне.

Доступ по IP может быть осуществлен с компьютера, на котором поднят KVM. Изнутри доступ происходит через NAT.

Возможные варианты настройки сети:

- NAT – это вариант по умолчанию. Внутренняя сеть, предоставляющая доступ к внешней сети с автоматическим применением NAT;
- Маршрутизация (Routed) – аналогично режиму NAT внутренняя сеть, предоставляющая доступ к внешней сети, но без NAT. Предполагает дополнительные настройки таблиц маршрутизации во внешней сети;
- Изолированная IPv4/IPv6 сеть (Isolated) – в этом режиме ВМ, подключенные к виртуальному коммутатору, могут общаться между собой и с хостом. При этом их трафик не будет выходить за пределы хоста;
- Bridge – подключение типа мост. Позволяет реализовать множество различных конфигураций, в том числе и назначение IP из реальной сети;

- SR-IOV pool (Single-root IOV) – перенаправление одной PCI сетевых карт хост-машины на ВМ. Технология SR-IOV повышает производительность сетевой виртуализации, избавляя гипервизор от обязанности организовывать совместное использование физического адаптера и перекладывая задачу реализации мультиплексирования на сам адаптер. В этом случае обеспечивается прямая пересылка ввода/вывода с ВМ непосредственно на адаптер.

3.5.2.1. Управление виртуальными сетями в командной строке

Команда просмотра списка виртуальных сетей:

```
# virsh net-list
Имя       Статус         Автозапуск          Persistent
----------------------------------------------------------------------------

default   активен             no                  yes
```

Просмотр информации для заданной виртуальной сети:

```
# virsh net-dumpxml <Имя сети>
```

Пример вывода этой команды (в формате XML):

```
# virsh net-dumpxml vnet1
<network connections='1'>
     <name>default</name>
     <uuid>54fdc7a0-b143-4307-a2f4-a9f9d997cb1b</uuid>
     <forward mode='nat'>
          <nat>
               <port start='1024' end='65535'/>
          </nat>
     </forward>
     <bridge name='virbr0' stp='on' delay='0'/>
     <mac address='52:54:00:3e:12:c7'/>
     <ip address='192.168.122.1' netmask='255.255.255.0'>
     <dhcp>
     <range start='192.168.122.2' end='192.168.122.254'/>
     </dhcp>
     </ip>
</network>
```

Другие команды управления виртуальными сетями:

- `virsh net-autostart имя_сети` – автоматический запуск заданной сети;
- `virsh net-create файл_XML` – создание и запуск новой сети на основе существующего XML-файла;
- `virsh net-define файл_XML` – создание нового сетевого устройства на основе существующего XML-файла (устройство не будет запущено);
- `virsh net-destroy имя_сети` – удаление заданной сети;
- `virsh net-name UUID_сети` – преобразование заданного идентификатора в имя сети;
- `virsh net-uuid имя_сети` – преобразование заданного имени в идентификатор UUID;
- `virsh net-start имя_неактивной_сети` – запуск неактивной сети;
- `virsh net-undefine имя_неактивной_сети` – удаление определения неактивной сети.

```
# virsh net-list –all
Имя       Статус         Автозапуск     Persistent
----------------------------------------------------------------------
default   не активен          no        yes
# virsh net-start default
# virsh net-list --all
Имя       Статус       Автозапуск      Persistent
----------------------------------------------------------------------
default   активен      no              yes
```

3.5.2.2. Управление виртуальными сетями в менеджере ВМ

В менеджере ВМ virt-manager существует возможность настройки виртуальных сетей для обеспечения сетевого взаимодействия ВМ как между собой, так и с хостовой ОС.

Для настройки виртуальной сети с помощью virt-manager необходимо:

1) в меню «Правка» выбрать «Свойства подключения» (рис. 15);
2) в открывшемся окне перейти на вкладку «Виртуальные сети» (рис. 16);

3) доступные виртуальные сети будут перечислены в левой части окна. Для доступа к настройкам сети необходимо выбрать сеть для доступа к ее настройкам.

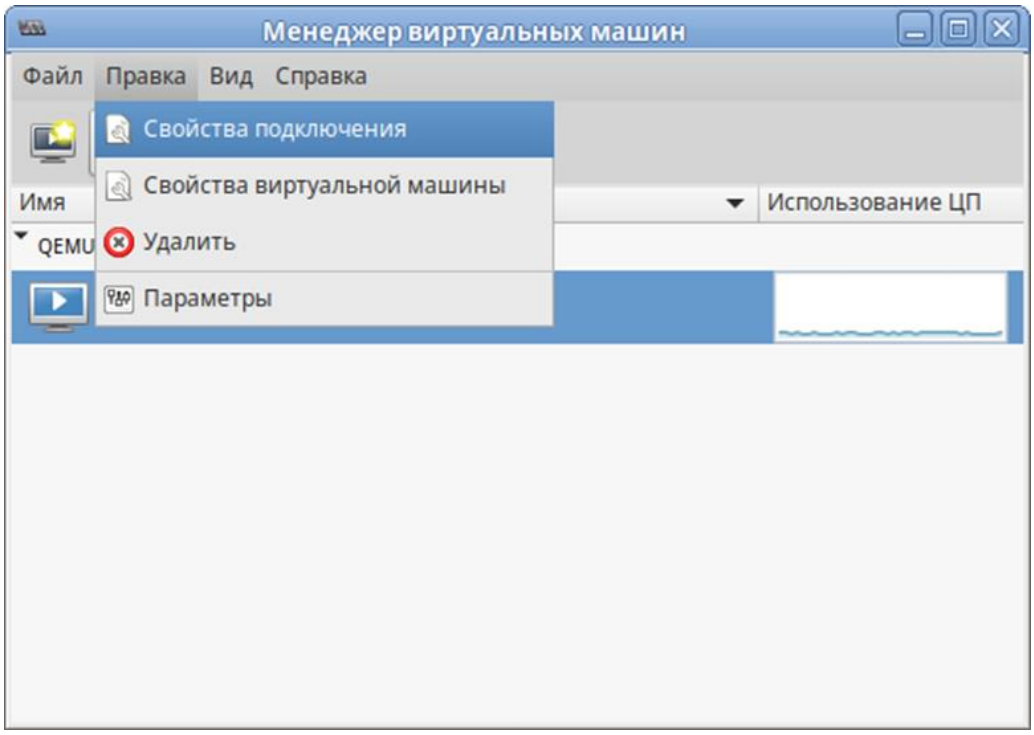

Рис. 15 – Меню «Правка»

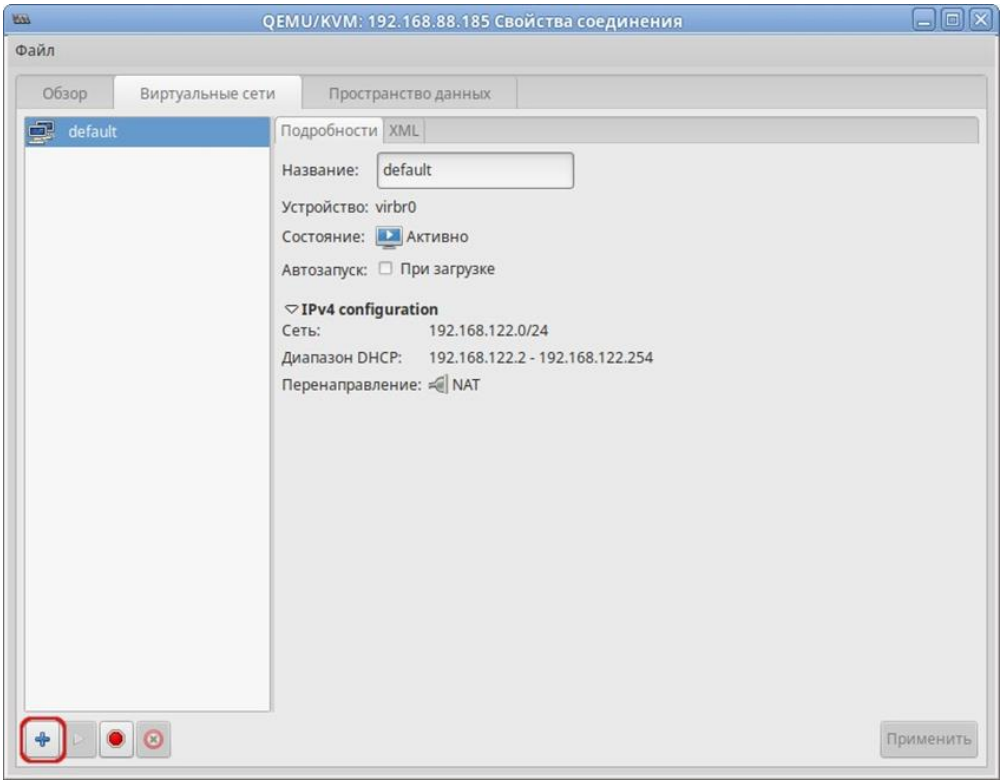

Рис. 16 – Окно параметров виртуальной сети

Для добавления новой виртуальной сети следует нажать на кнопку «Добавить сеть»  (рис. 16), расположенную в нижнем левом углу диалогового окна «Свойства соединения». В открывшемся окне (рис. 17) следует ввести имя для новой сети и задать необходимые настройки: выбрать способ подключения виртуальной сети к физической, ввести пространство адресов IPv4 для виртуальной сети, указать диапазон DHCP, задав начальный и конечный адрес и нажать на кнопку «Готово».

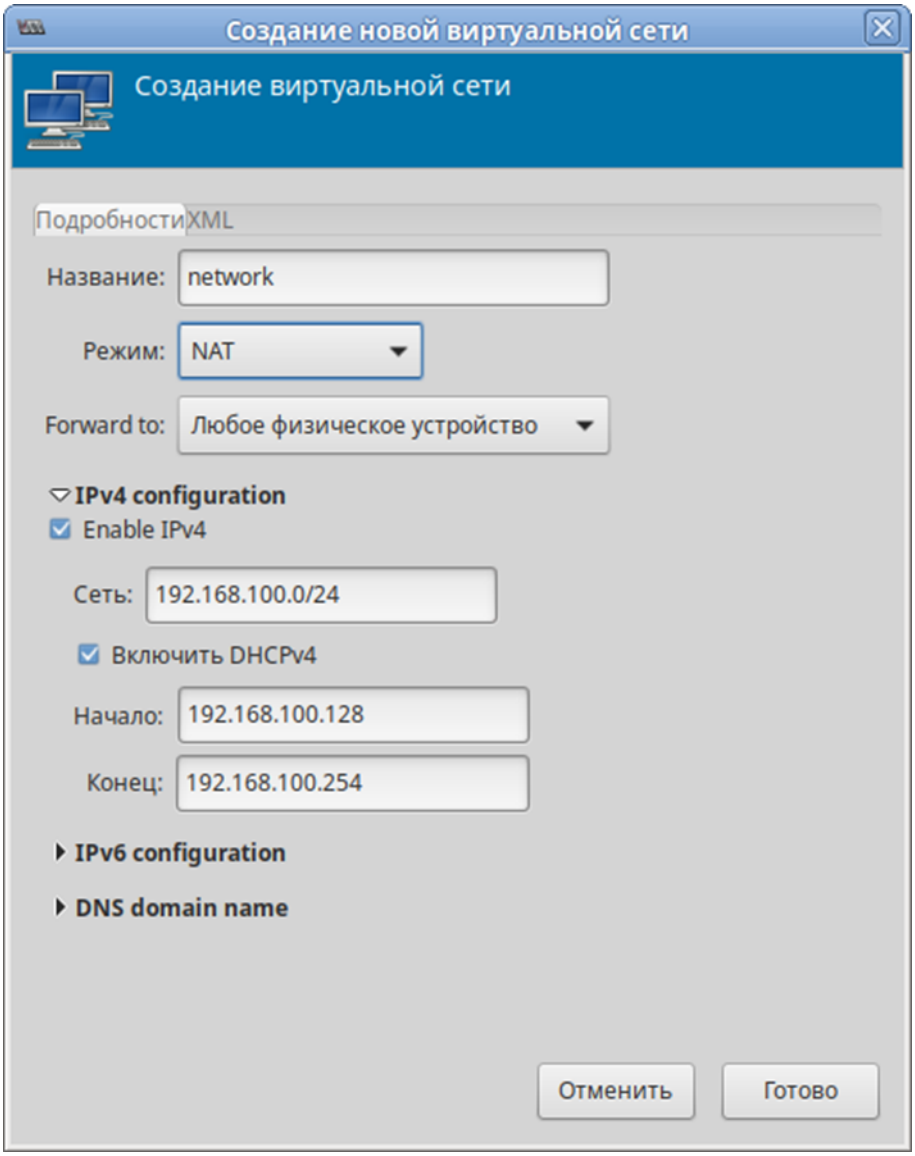

Рис. 17 – Создание новой виртуальной сети

3.5.3. Управление хранилищами

API-интерфейс libvirt обеспечивает удобную абстракцию для размещения образов ВМ и файловых систем, который носит название `storage pools` (пул хранилищ). Пул хранилищ – это локальный каталог, локальное устройство хранения данных (физический диск, логический том или хранилище на основе хост-адаптера шины SCSI [SCSI HBA]), файловая система NFS (network file system), либо сетевое хранилище блочного уровня, управляемое посредством libvirt и позволяющее создать и хранить один или более образов ВМ.

По умолчанию команды на базе libvirt используют в качестве исходного пула хранилищ для каталога файловой системы каталог `/var/lib/libvirt/images` на хосте виртуализации. Новый пул хранилищ можно с легкостью создать с помощью команды `virsh pool-create-as`.

Образ диска – это снимок данных диска ВМ, сохраненный в том или ином формате. libvirt понимает несколько форматов образов. Так же возможна работа с образами CD/DVD дисков. Каждый образ хранится в том или ином хранилище.

Типы хранилищ, с которыми работает libvirt:

- `dir` – каталог в файловой системе;
- `disk` – физический диск;
- `fs` – отформатированное блочное устройство;
- `gluster` – файловая система Gluster;
- `isci` – хранилище iSCSI;
- `logical` – группа томов LVM;
- `mpath` – регистратор многопутевых устройств;
- `netfs` – экспорт каталога из сети;
- `rbd` – блочное устройство RADOS/Ceph;
- `scsi` – хост-адаптер SCSI;
- `sheepdog` – файловая система Sheepdog;
- `zfs` – пул ZFS.

3.5.3.1. Управление хранилищами в командной строке

Новый пул хранилищ можно создать с помощью команды `virsh pool-create-as`. Например, следующая команда демонстрирует обязательные аргументы, которые необходимо указать при создании пула хранилищ на основе NFS (netfs):

```
# virsh pool-create-as NFS-POOL netfs \
--source-host 192.168.88.180 \
--source-path /export/storage \
--target /var/lib/libvirt/images/NFS-POOL
```

Первый аргумент (NFS-POOL) идентифицирует имя нового пула хранилищ, второй аргумент идентифицирует тип создаваемого пула хранилищ. Аргумент опции `--source-host` идентифицирует хост, который экспортирует каталог пула хранилищ посредством NFS. Аргумент опции `--source-path` определяет имя экспортируемого каталога на этом хосте. Аргумент опции `--target` идентифицирует локальную точку монтирования, которая будет использоваться для обращения к пулу хранилищ.

Примечание. Для возможности монтирования NFS хранилища необходимо запустить службы `rpcbind` и `nfslock`:

```
# systemctl start rpcbind
# systemctl start nfslock
```

После создания нового пула хранилищ он будет указан в выходной информации команды `virsh pool-list`:

```
# virsh pool-list --all –details
Имя       Статус   Автозапуск Persistent Capacity  Распределение Доступно
-------------------------------------------------------------------------------------------
default   работает yes        yes        17,46 GiB 12,32GiB      5,15 GiB
NFS-POOL работает no         no         29,40 GiB 6,08 GiB      23,33 GiB
```

В выводе команды видно, что опция «Автозапуск» («Autostart») для пула хранилищ NFS- POOL имеет значение no (нет), т. е. после перезапуска системы этот пул не будет автоматически доступен для использования, и что опция «Persistent» также имеет значение «no», т. е. после перезапуска системы этот пул вообще не будет определен. Пул хранилищ является постоянным только в том случае, если он сопровождается XML-описанием пула хранилищ, которое находится в каталоге

`/etc/libvirt/storage`. XML-файл описания пула хранилищ (файл с расширением xml) имеет такое же имя, как у пула хранилищ, с которым он ассоциирован.

Чтобы создать файл XML-описания для сформированного в ручном режиме пула хранилищ, следует воспользоваться командой `virsh pool-dumpxml`, указав в качестве ее заключительного аргумента имя пула, для которого нужно получить XML-описание. Эта команда осуществляет запись в стандартное устройство вывода, поэтому необходимо перенаправить ее выходную информацию в соответствующий файл.

Например, следующая команда создаст файл XML-описания для созданного ранее пула хранилищ NFS-POOL:

```
# virsh pool-dumpxml NFS-POOL > /etc/libvirt/storage/NFS-POOL.xml
```

Чтобы задать для пула хранилищ опцию «Автозапуск» («Autostart»), можно воспользоваться командой `virsh pool-autostart`:

```
# virsh pool-autostart NFS-POOL
Добавлена метка автоматического запуска пула NFS-POOL
```

Маркировка пула хранилищ как «Autostart» говорит о том, что этот пул хранилищ будет доступен после любого перезапуска хоста виртуализации (каталог `/etc/libvirt/storage/autostart` будет содержать символьную ссылку на XML-описание этого пула хранилищ).

3.5.3.2. Настройка пулов хранилищ в менеджере ВМ

Для настройки пулов хранилищ с помощью virt-manager необходимо:

1) в меню «Правка» выбрать «Свойства подключения» (рис. 18);

2) в открывшемся окне перейти на вкладку «Пространство данных» (рис. 19).

Для добавления пула следует нажать на кнопку «Добавить пул»  , расположенную в нижнем левом углу диалогового окна «Свойства соединения» (см. рис. 19). В открывшемся окне (рис. 20) следует выбрать тип пула, на втором шаге (рис. 21) задаются параметры пула.

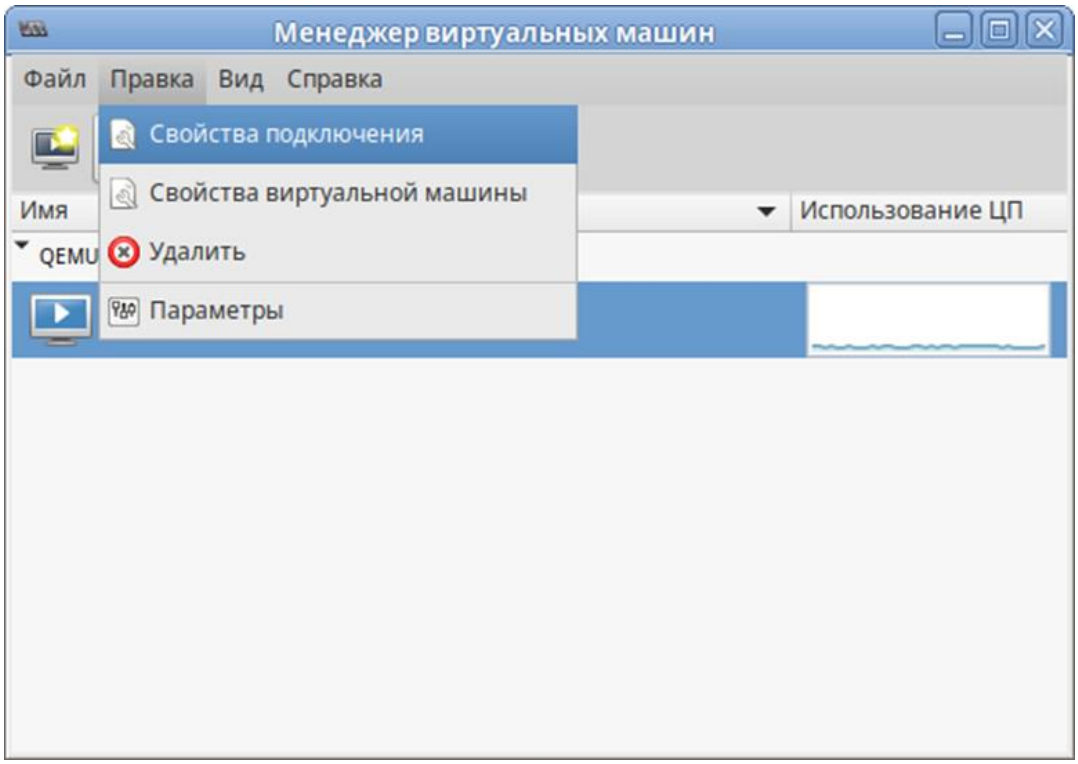

Рис. 18 – Меню «Правка»

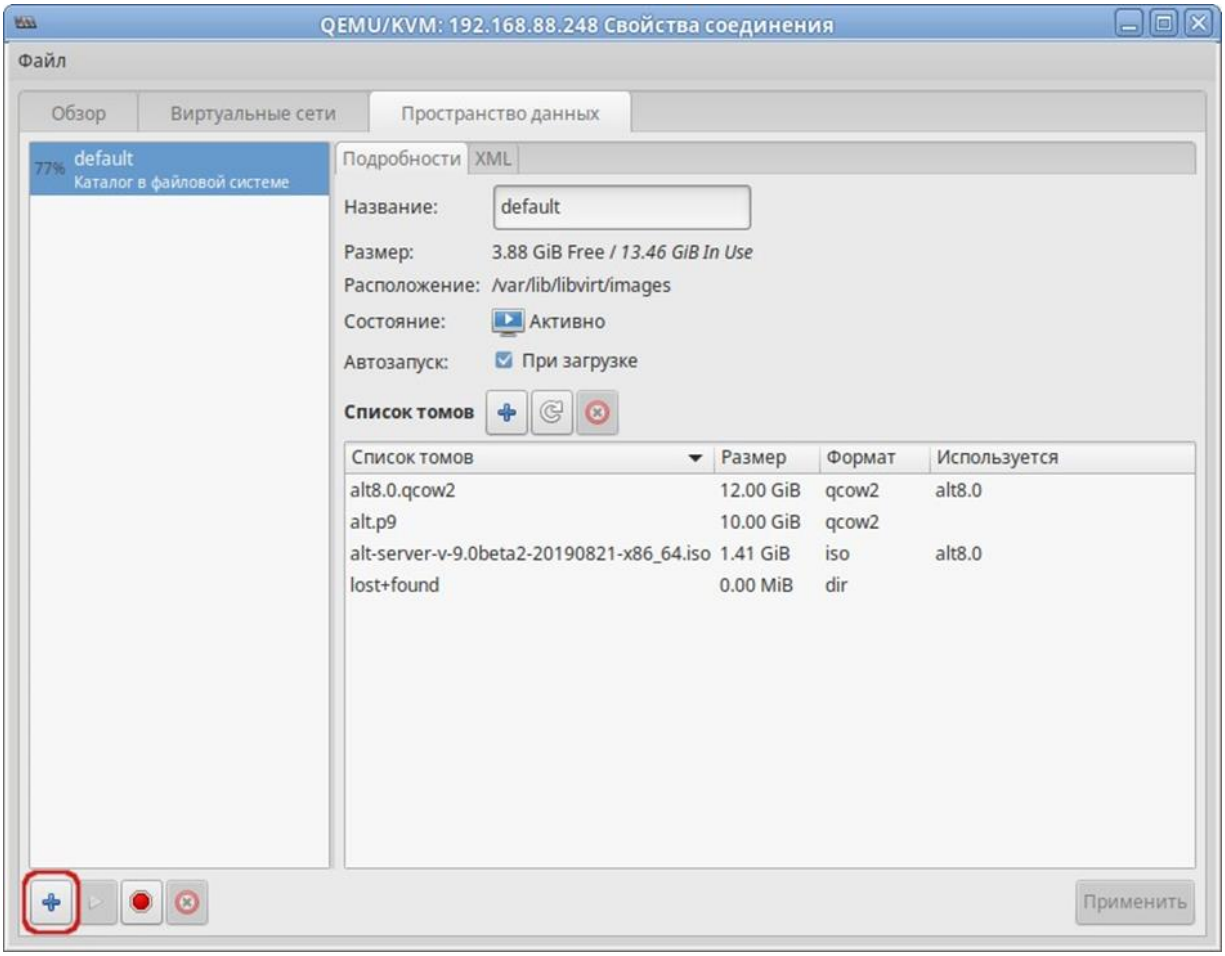

Рис. 19 – Вкладка «Пространство данных»

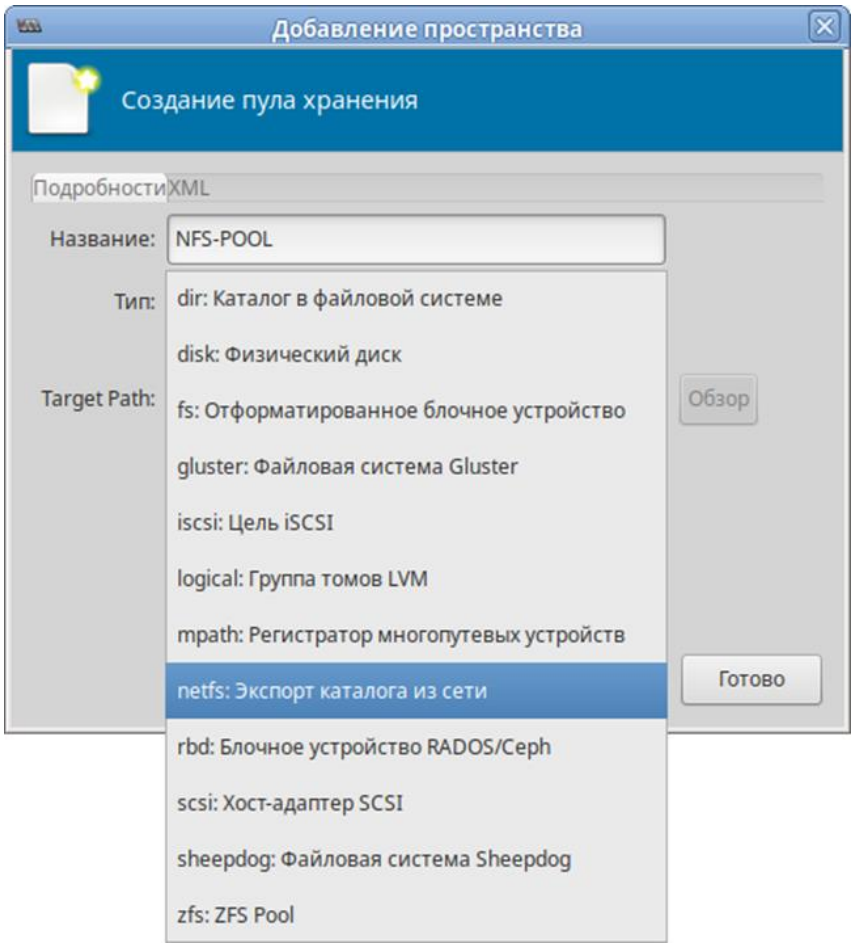

Рис. 20 – Создание пула хранения. Выбор типа пула

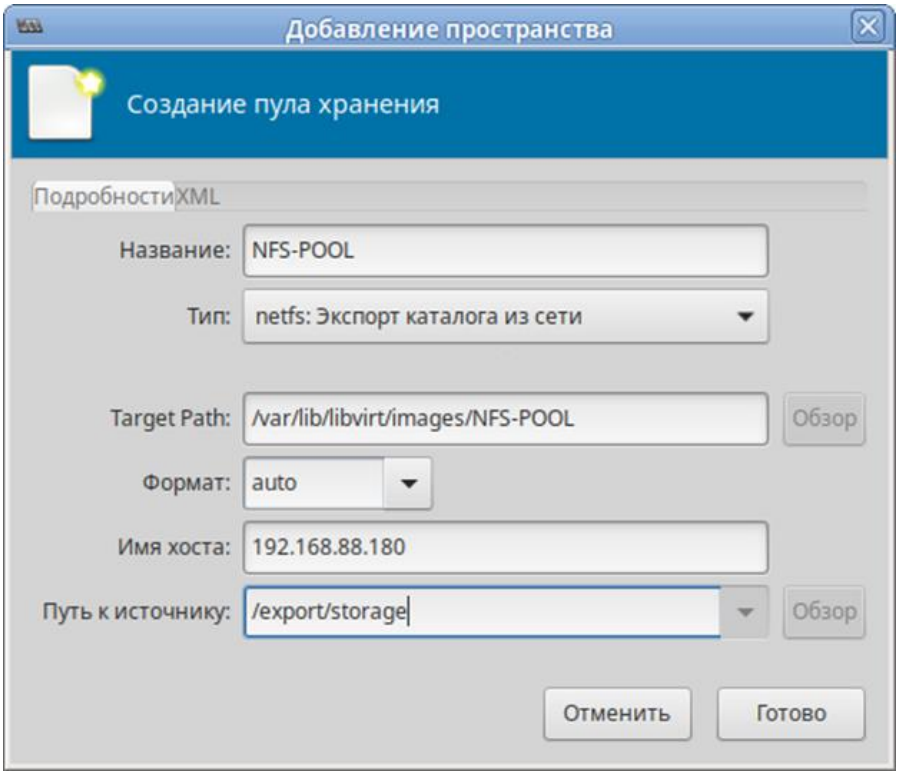

Рис. 21 – Создание пула хранения. Ввод параметров

3.6. Запуск и управление функционированием ВМ

3.6.1. Управление состоянием ВМ в командной строке

Команды управления состоянием ВМ:

- `start` – запуск ВМ;
- `shutdown` – завершение работы. Поведение выключаемой ВМ можно контролировать с помощью параметра `on_shutdown` (в файле конфигурации);
- `destroy` – принудительная остановка. Использование `virsh destroy` может повредить гостевые файловые системы. Рекомендуется использовать опцию `shutdown`;
- `reboot` – перезагрузка ВМ. Поведение перезагружаемой ВМ можно контролировать с помощью параметра `on_reboot` (в файле конфигурации);
- `suspend` – приостановить ВМ. Когда ВМ находится в приостановленном состоянии, она потребляет системную оперативную память, но не ресурсы процессора;
- `resume` – возобновить работу приостановленной ВМ;
- `save` – сохранение текущего состояния ВМ. Эта команда останавливает ВМ, сохраняет данные в файл, что может занять некоторое время (зависит от объема ОЗУ ВМ);
- `restore` – восстановление ВМ, ранее сохраненной с помощью команды `virsh save`. Сохраненная машина будет восстановлена из файла и перезапущена (это может занять некоторое время). Имя и идентификатор UUID ВМ останутся неизменными, но будет предоставлен новый идентификатор домена;
- `undefine` – удалить ВМ (конфигурационный файл тоже удаляется);
- `autostart` – добавить ВМ в автозагрузку;
- `autostart --disable` – удалить из автозагрузки.

В результате выполнения следующих команд, ВМ `alt-server` будет остановлена и затем удалена:

```
# virsh -c qemu:///system destroy alt-server
# virsh -c qemu:///system undefine alt-server
```

3.6.2. Управление состоянием ВМ в менеджере ВМ

Для запуска ВМ в менеджере ВМ virt-manager, необходимо выбрать ВМ из списка и нажать на кнопку «Включить ВМ» (рис. 22).

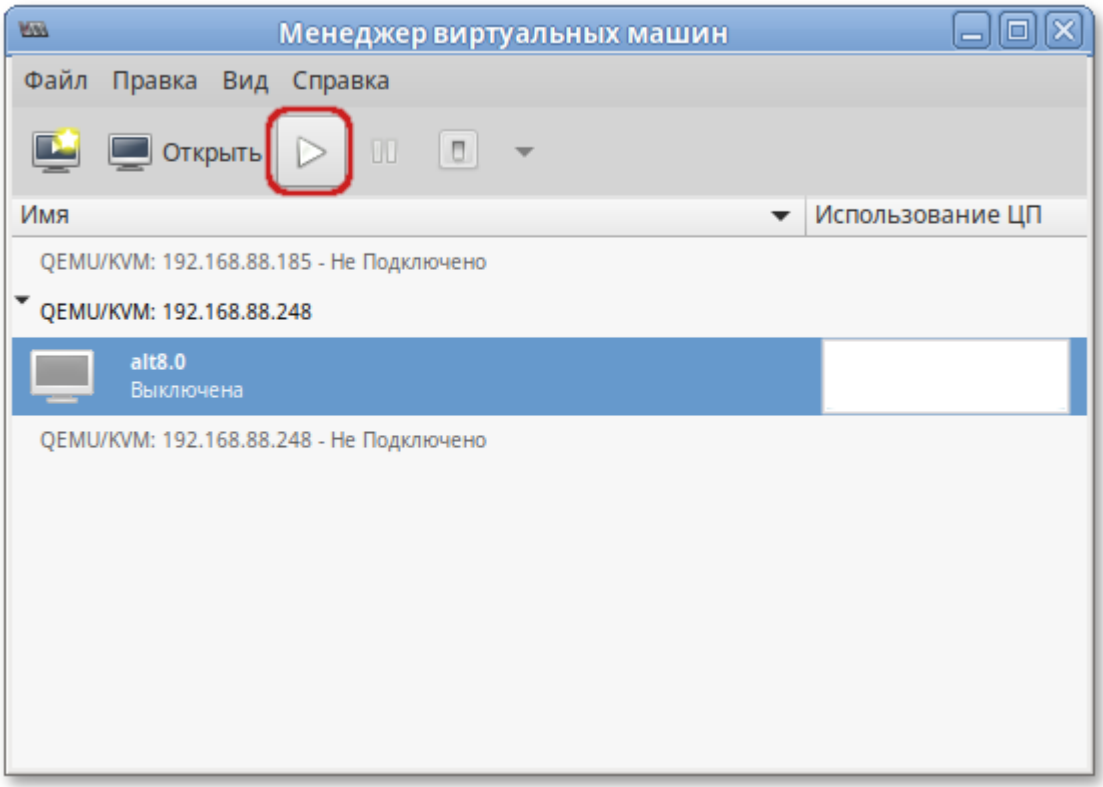

Рис. 22 – Включение ВМ

Для управления запущенной ВМ используются соответствующие кнопки панели инструментов virt-manager (рис. 23).

Управлять состоянием ВМ можно также выбрав соответствующий пункт в контекстном меню ВМ (рис. 24).

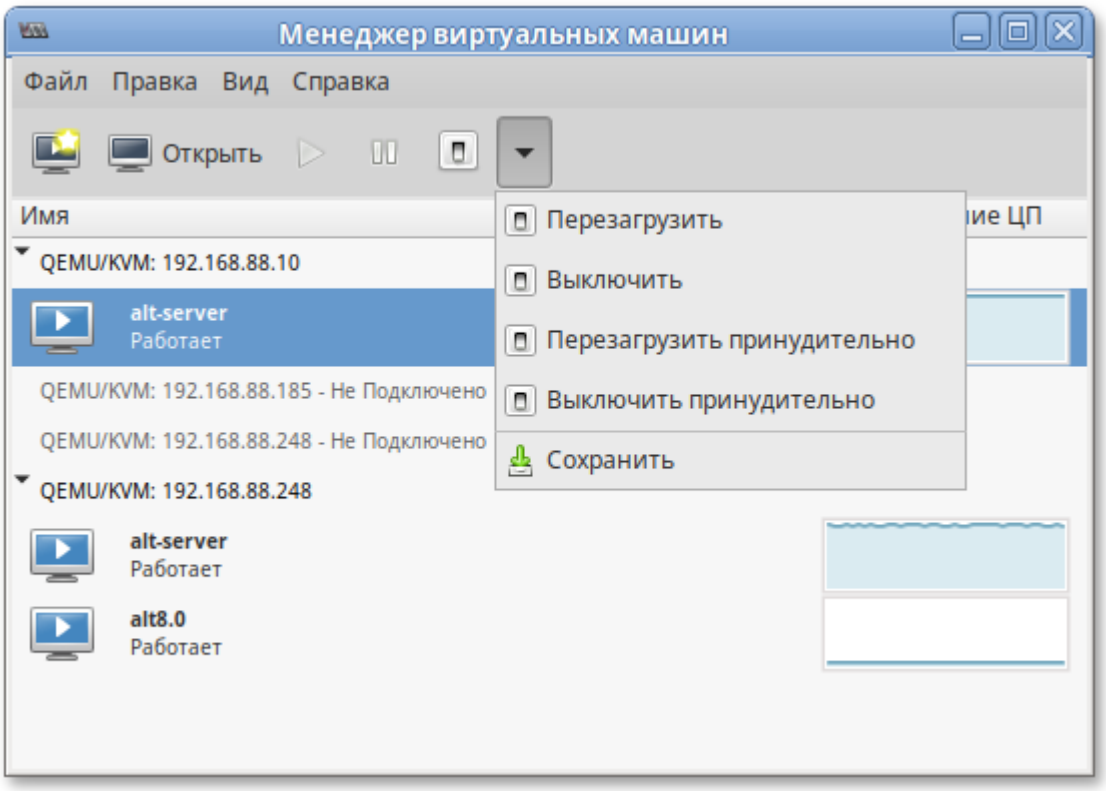

Рис. 23 – Кнопки управления состоянием ВМ

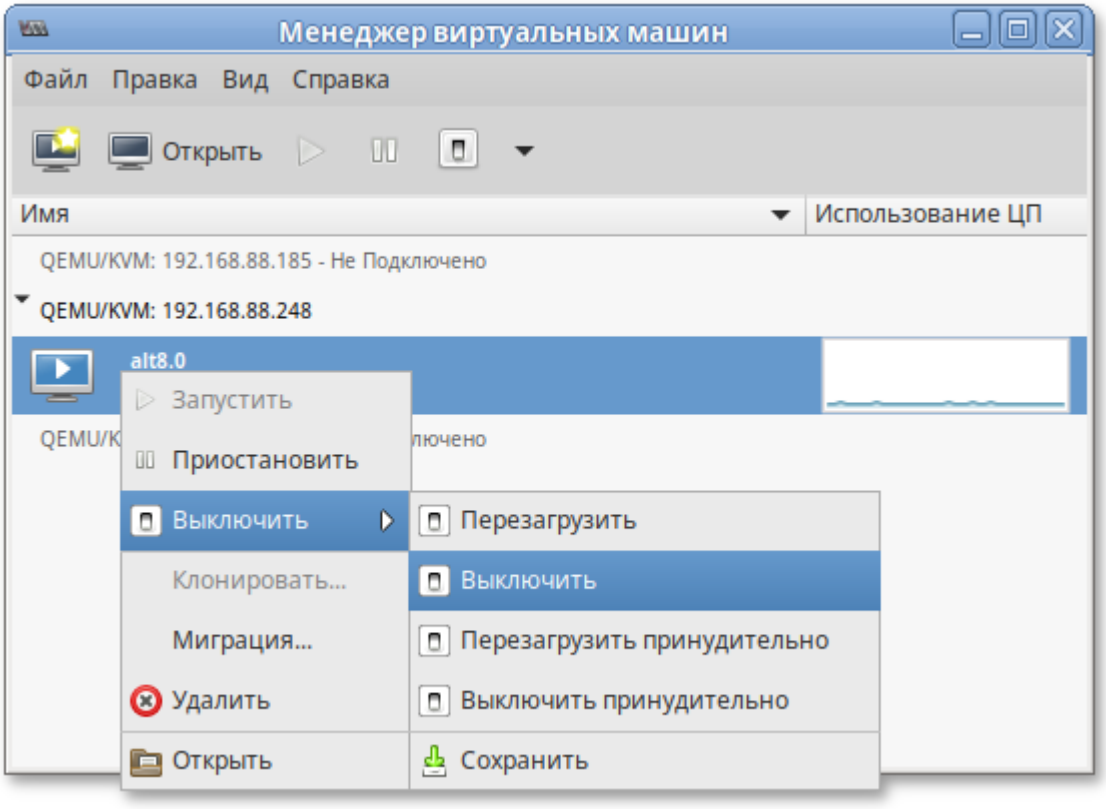

Рис. 24 – Контекстное меню ВМ

3.7. Миграция ВМ

Под миграцией понимается процесс переноса ВМ с одного узла на другой.

Живая миграция позволяет перенести работу ВМ с одного физического хоста на другой без остановки ее работы.

Для возможности миграции ВМ, ВМ должна быть создана с использованием общего пула хранилищ (NFS, iSCSI, GlusterFS, CEPH).

Примечание. Живая миграция возможна даже без общего хранилища данных (с опцией `--copy-storage-all`). Но это приведет к большому трафику при копировании образа ВМ между серверами виртуализации и к заметному простою сервиса. Чтобы миграция была по-настоящему «живой» с незаметным простоем необходимо использовать общее хранилище.

3.7.1. Миграция с помощью virsh

ВМ можно перенести на другой узел с помощью команды `virsh`. Для выполнения живой миграции нужно указать параметр `--live`. Команда переноса:

```
# virsh migrate --live VMName DestinationURL
```

где:

- `VMName` – имя перемещаемой ВМ;
- `DestinationURL` – URL или имя хоста узла назначения. Узел назначения должен использовать тот же гипервизор, и служба libvirt на нем должна быть запущена.

После ввода команды будет запрошен пароль администратора узла назначения.

Для выполнения живой миграции ВМ, например, `alt8.0` на узел `192.168.88.190` с помощью утилиты virsh, необходимо выполнить следующие действия:

1) убедиться, что ВМ запущена:

```
# virsh list
 ID    Имя          Статус
----------------------------------------
 1     alt8.0       работает
```

2) выполнить команду, чтобы начать перенос ВМ на узел `192.168.88.190` (после ввода команды будет запрошен пароль пользователя `root` системы назначения):

```
# virsh migrate --live alt8.0 qemu+ssh://192.168.88.190/system
```

3) процесс миграции может занять некоторое время в зависимости от нагрузки и размера ВМ. `virsh` будет сообщать только об ошибках. ВМ будет продолжать работу на исходном узле до завершения переноса;

4) проверить результат переноса – выполнить на узле назначения команду:

```
# virsh list
 ID    Имя         Статус
----------------------------------------------------
 1     alt8.0      работает
```

3.7.2. Миграция с помощью virt-manager

Менеджер ВМ virt-manager поддерживает возможность миграции ВМ между серверами виртуализации.

Для выполнения миграции, в virt-manager необходимо выполнить следующие действия:

1) подключить второй сервер виртуализации («Файл» → «Добавить соединение…»);
2) в контекстном меню ВМ (она должна быть запущена) (рис. 25) выбрать пункт «Миграция»;
3) в открывшемся окне (рис. 26) выбрать конечный узел и нажать на кнопку «Миграция».

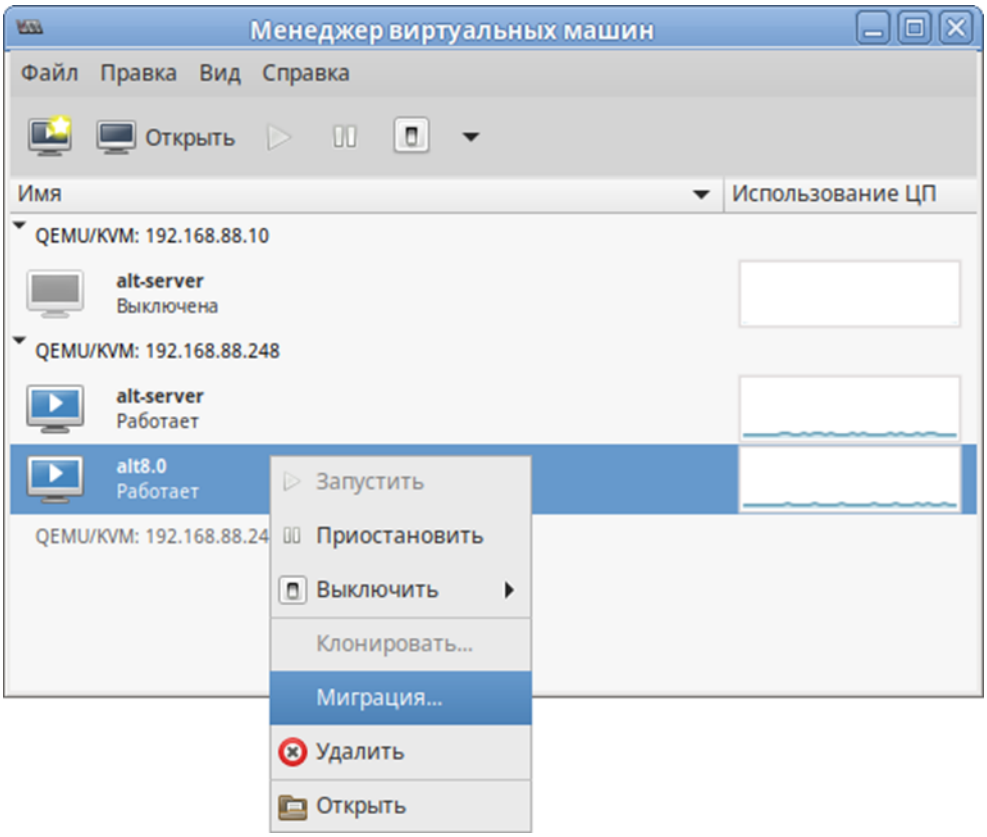

Рис. 25 – Пункт «Миграция» в контекстном меню ВМ

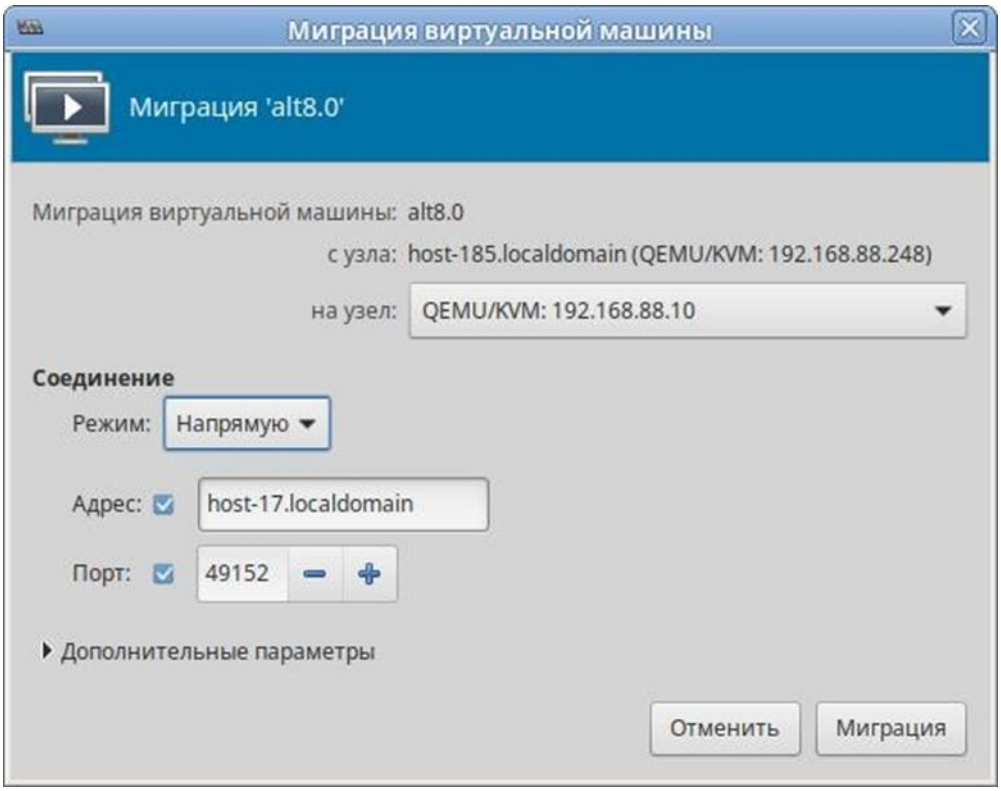

Рис. 26 – Миграция ВМ

При этом конфигурационный файл перемещаемой машины не переходит на новый узел, поэтому при ее выключении она вновь появится на старом хосте. В связи с этим, для совершения полной живой миграции, при котором конфигурация ВМ будет перемещена на новый узел, необходимо воспользоваться утилитой командной строки `virsh`:

```
# virsh migrate --live --persistent --undefinesource \ alt8.0 qemu+ssh://192.168.88.10/system
```

3.8. Снимки машины

Примечание. Снимок (snapshot) текущего состояния машины можно создать только если виртуальный жесткий диск в формате `*.qcow2`.

3.8.1. Управления снимками ВМ в консоли

Команда создания снимка (ОЗУ и диск) из файла XML:

```
# virsh snapshot-create <domain> [--xmlfile <строка>] [--disk-only] [-- live]...
```

Команда создания снимка (ОЗУ и диск) напрямую из набора параметров:

```
# virsh snapshot-create-as <domain> [--name <строка>] [--disk-only] [-- live]...
```

Пример создания снимка ВМ:

```
# virsh snapshot-create-as --domain alt-server --name 28nov2019
Снимок домена 28nov2019 создан
```

где:

- `alt-server` – имя ВМ;
- `28nov2019` – название снимка.

После того, как снимок ВМ будет сделан, резервные копии файлов конфигураций будут находиться в каталоге `/var/lib/libvirt/qemu/snapshot/`.

Пример создания снимка диска ВМ:

```
# virsh snapshot-create-as --domain alt-server --name 05dec2019 -diskspec vda,file=/var/lib/libvirt/images/sn1.qcow2 --disk-only --atomic
Снимок домена 05dec2019 создан
```

Просмотр существующих снимков для домена `alt-server`:

```
# virsh snapshot-list --domain alt-server
 Имя            Время создания                   Статус
-------------------------------------------------------------------------------
28nov2019        2019-11-28 08:50:05    +0200    running
05dec2019        2019-12-05 13:14:11    +0200    disk-snapshot
```

Восстановить ВМ из снимка:

```
# virsh snapshot-revert --domain alt-server --snapshotname 28nov2019 – running
```

Удалить снимок:

```
# virsh snapshot-delete --domain alt-server --snapshotname 28nov2019
```

3.8.2. Управления снимками ВМ virt-manager

Для управления снимками ВМ в менеджере ВМ virt-manager, необходимо:

1) в главном окне менеджера выбрать ВМ;
2) нажать на кнопку «Открыть»;
3) в открывшемся окне нажать на кнопку «Управление снимками» (рис. 27). Появится окно управления снимками ВМ.

Для создания нового снимка следует нажать на кнопку «Создать новый снимок»  , расположенную в нижнем левом углу окна управления снимками ВМ. В открывшемся окне (рис. 28) следует указать название снимка и нажать на кнопку «Готово».

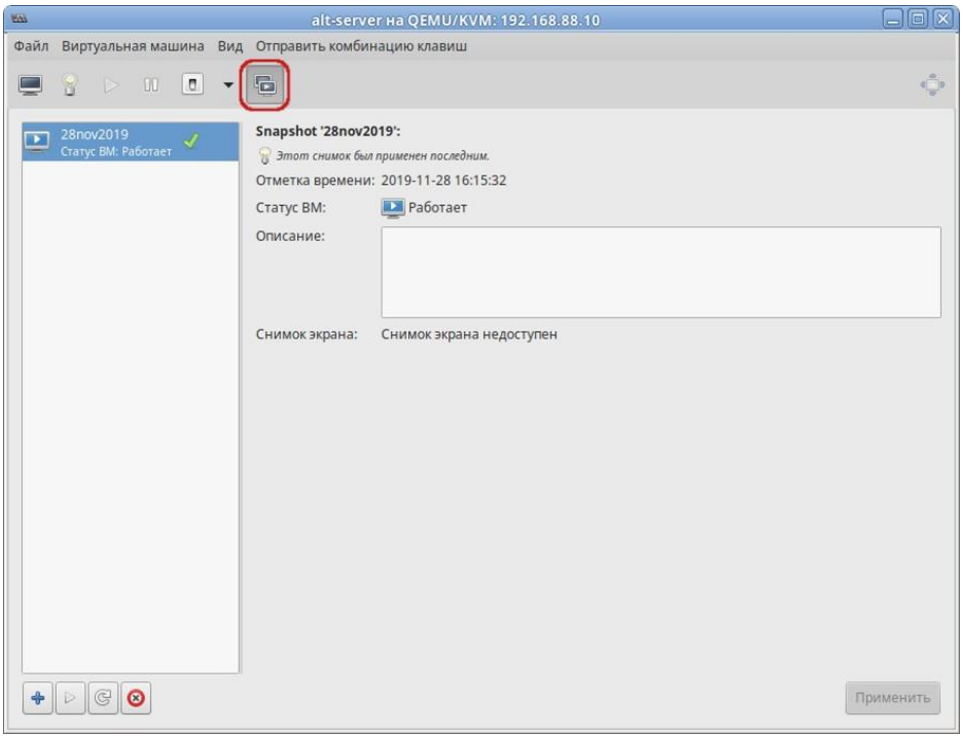

Рис. 27 – Управление снимками ВМ

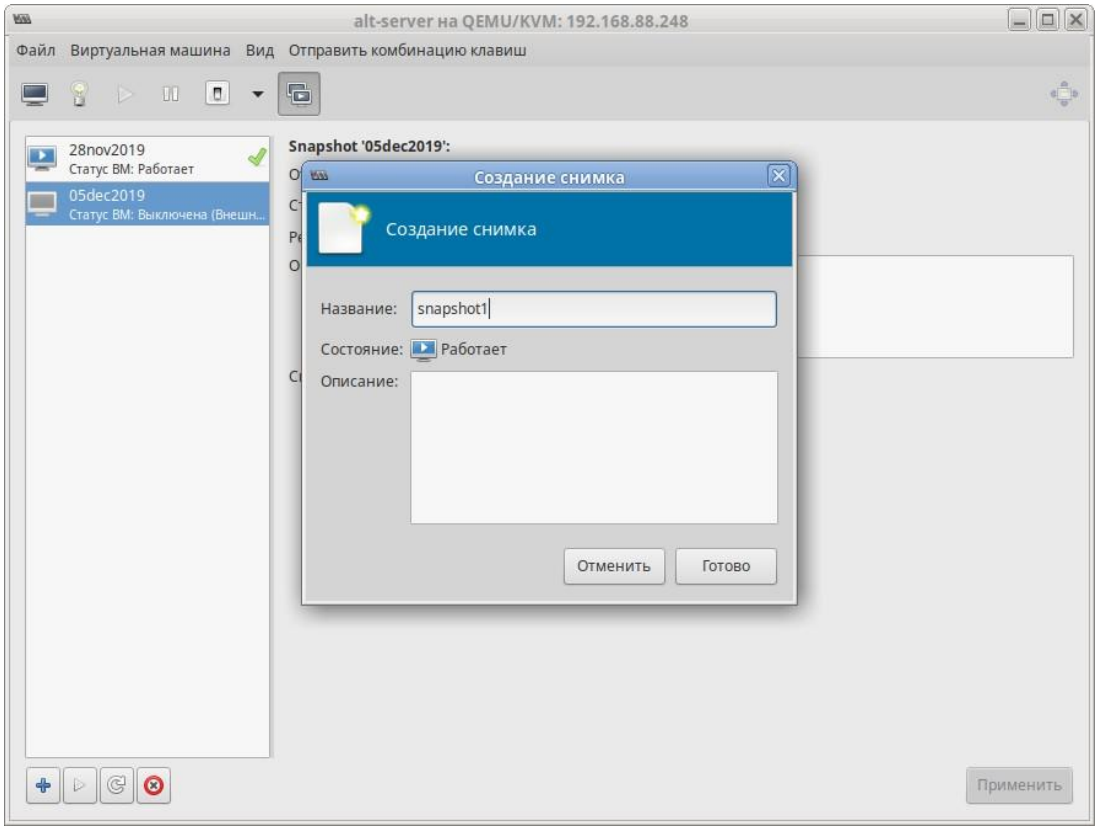

Рис. 28 – Создание снимка

Для того чтобы восстановить ВМ из снимка или удалить снимок, следует воспользоваться контекстным меню снимка (рис. 29).

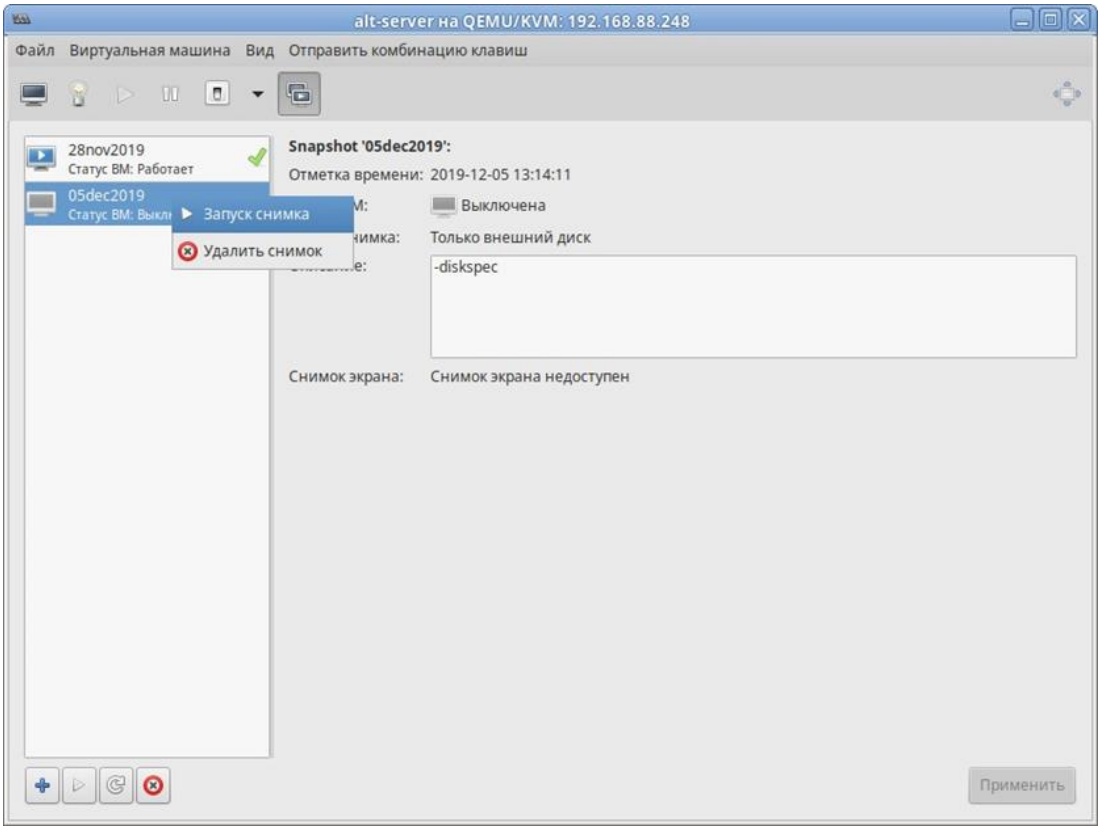

Рис. 29 – Контекстное меню снимка

3.9. Управление доступом в виртуальной инфраструктуре

Права пользователя могут управляться с помощью правил `polkit`.

В каталоге `/usr/share/polkit-1/actions/` имеются два файла с описанием возможных действий для работы с ВМ, предоставленные разработчиками libvirt:

- файл `org.libvirt.unix.policy` описывает мониторинг ВМ и управление ими;
- в файле `org.libvirt.api.policy` перечислены конкретные действия (остановка, перезапуск и т. д.), которые возможны, если предыдущая проверка пройдена.

Перечисление конкретных свойств с комментариями доступно в файле `/usr/share/polkit-1/actions/org.libvirt.api.policy`.

Например, действие "Manage local virtualized systems" в файле org.libvirt.unix.policy:

```
<action id="org.libvirt.unix.manage">
    <description>Manage local virtualized systems</description>
    <message>System policy prevents management of local virtualized systems</message>
    <defaults>
        <allow_any>auth_admin_keep</allow_any>
        <allow_inactive>auth_admin_keep</allow_inactive>
        <allow_active>auth_admin_keep</allow_active>
    </defaults>
</action>
```

В libvirt названия объектов и разрешений отображаются в имена polkit действий, по схеме:

```
org.libvirt.api.$объект.$разрешение
```

Например, разрешение search-storage-vols на объекте storage_pool отображено к действию polkit:

```
org.libvirt.api.storage-pool.search-storage-vols
```

Libvirt применяет контроль доступа ко всем основным типам объектов в его API. В таблице 4 приведены объекты, со своими наборами разрешений.

Т а б л и ц а 4 – Типы разрешений к объектам libvirt

Объект	Разрешения	Описание
Connect	detect-storage-pools	Обнаружение хранилищ
	getattr	Подключение
	interface-transaction	Операции с интерфейсом
	pm-control	Управление питанием
	read	Просмотр
	search-domains	Список доменов
	search-interfaces	Список интерфейсов
	search-networks	Список сетей
	search-node-devices	Список узлов
	search-nwfilters	Список сетевых фильтров
	search-secrets	List secrets
	search-storage-pools	Список хранилищ
	write	Изменение

Продолжение таблицы 4

Объект	Разрешения	Описание
Domain	block-read	Read domain block
	mem-read	Просмотр памяти
	migrate	Миграция
	open-device	Open device
	open-graphics	Open graphics
	open-namespace	Open namespace
	inject-nmi	Inject domain NMI
	pm-control	Управление питанием
	read	Чтение
	read-secure	Защищенный просмотр
	reset	Перезагрузить
	save	Сохранить
	screenshot	Получить screenshot
	send-input	Send input
	send-signal	Send signal
	set-password	Установка паролей
	set-time	Установка времени
	snapshot	Снимок
	start	Запуск
	stop	Останов
	suspend	Приостановка
	write	Изменение
	hibernate	Hibernate domain
	init-control	Domain init control
Interface	delete	Удаление
	getattr	Доступ
	read	Просмотр
	save	Сохранение
	start	Запуск
	stop	Останов
	write	Изменение
Network	delete	Удаление
	getattr	Доступ
	read	Просмотр
	save	Сохранение
	start	Запуск
	stop	Останов
	write	Изменение
NodeDevice	detach	Отсоединение
	getattr	Доступ
	read	Просмотр

Окончание таблицы 4

Объект	Разрешения	Описание
	start	Запуск
	stop	Останов
	write	Изменение
NWFilter	delete	Удаление
	getattr	Доступ
	read	Просмотр
	save	Сохранение
	write	Изменение
Secret	delete	Удаление
	getattr	Доступ
	read	Просмотр
	read-secure	Безопасный просмотр
	save	Save secret
	write	Write secret
StoragePool	delete	Удаление
	refresh	Обновление
	format	Форматирование
	getattr	Доступ
	read	Просмотр
	save	Сохранение
	search-storage-vols	Список томов
	start	Запуск
	stop	Останов
	write	Изменение
	create	Создание
StorageVol	data-read	Просмотр данных
	data-write	Запись данных
	delete	Удаление
	format	Форматирование
	getattr	Доступ
	read	Просмотр
	resize	Изменение размера

Чтобы определить правила авторизации, polkit должен однозначно определить объект. Libvirt предоставляет ряд атрибутов для определения объектов при выполнении проверки прав доступа. Набор атрибутов изменяется в зависимости от типа объекта (таблица 5).

Т а б л и ц а 5 – Атрибуты объектов libvirt

Объект	Атрибут	Описание
Connect	connect_driver	Название подключения
Domain	connect_driver	Название подключения
	domain_name	Название домена, уникально для локального хоста
	domain_uuid	UUID домена, уникально
Interface	connect_driver	Название подключения
	interface_name	Название сетевого интерфейса, уникально для локального хоста
	interface_macaddr	MAC-адрес сетевого интерфейса, не уникальный глобально
Network	connect_driver	Название подключения
	network_name	Название сети, уникально для локального хоста
	network_uuid	UUID сети, уникально
NodeDevice	connect_driver	Название подключения
	node_device_name	Название устройства, уникально для локального хоста
NWFilter	connect_driver	Название подключения
	nwfilter_name	Название сетевого фильтра, уникально для локального хоста
	nwfilter_uuid	UUID сетевого фильтра, уникально
Secret	connect_driver	Название подключения
	secret_uuid	UUID уникально
	secret_usage_volume	Название тома
	secret_usage_ceph	Название Ceph сервера
	secret_usage_target	Название iSCSI
	secret_usage_name	Название TLS
StoragePool	connect_driver	Название подключения
	pool_name	Название хранилища, уникально для локального хоста
	pool_uuid	UUID хранилища, уникально

По умолчанию для запуска virt-manager требуется ввод пароля пользователя с идентификатором `root`. Для того, чтобы virt-manager запускался от нужного пользователя (в примере – `test`), необходимо добавить этого пользователя в группу `vmusers` и перелогиниться, при необходимости перезапустить libvirt, polkit, nscd:

```
# gpasswd -a test vmusers
# service libvirtd restart
# service polkit restart
# service nscd restart
```

Добавить в файл `/etc/libvirt/libvirtd.conf` строку:

```
access_drivers = [ "polkit" ]
```

Перезапустить libvirt:

```
# service libvirtd restart
```

3.9.1. Пример тонкой настройки

Есть две ВМ: `alt1`, `alt2`. Необходимо разрешить пользователю `test` (должен быть в группе `vmusers`) действия только с доменом `alt1`. Для этого необходимо выполнить следующие действия:

1) раскомментировать в файле `/etc/libvirt/libvirtd.conf` строку: `access_drivers = [ "polkit" ]`
2) перезапустить libvirt: `# systemctl restart libvirtd`
3) создать файл `/etc/polkit-1/rules.d/100-libvirt-acl.rules` (имя произвольно) следующего вида:

```
==========================
polkit.addRule(function(action, subject) {
// разрешить пользователю test действия с доменом "alt1"
if (action.id.indexOf("org.libvirt.api.domain.") ==0 &&
subject.user == "test") {
        if (action.lookup("domain_name") == 'alt1') {
return polkit.Result.YES;
}
else { return polkit.Result.NO; }
}
else {
// разрешить пользователю test действия с
//подключениями, хранилищем и прочим
if (action.id.indexOf("org.libvirt.api.") == 0 &&
subject.user == "test") {
polkit.log("org.libvirt.api.Yes");
return polkit.Result.YES;
}
else { return polkit.Result.NO; }
}})
==============================
```

4) выйти и снова войти в ОС.

В результате выполненных действий пользователю `test` машина `alt1` видна, а машина `alt2` – нет.

Права можно настраивать более тонко, например, разрешив пользователю `test` запускать ВМ, но запретить ему все остальные действия с ней, для этого надо разрешить действие `org.libvirt.api.domain.start`:

```
==========================
polkit.addRule(function(action, subject) {
     // разрешить пользователю test только запускать ВМ в
     // домене "alt1"
     if (action.id. == "org.libvirt.api.domain.start") &&
           subject.user == "test") {
               if (action.lookup("domain_name") == 'alt1') {
                    return polkit.Result.YES;
               }
               else { return polkit.Result.NO; }
}
     });
==========================
```

Предоставить право запускать ВМ, только пользователям группы `wheel`:

```
if (action.id == "org.libvirt.api.domain.start") {
     if (subject.isInGroup("wheel")) {
          return polkit.Result.YES;
     } else {
          return polkit.Result.NO;
     }
};
```

Предоставить право останавливать ВМ, только пользователям группы `wheel`:

```
if (action.id == "org.libvirt.api.domain.stop") {
     if (subject.isInGroup("wheel")) {
          return polkit.Result.YES;
     } else {
          return polkit.Result.NO;
     }
};
```

Можно также вести файл журнала, используя правила polkit. Например, делать запись в журнал при старте ВМ:

```
if (action.id.match("org.libvirt.api.domain.start") ) {
     polkit.log("action=" + action);
     polkit.log("subject=" + subject);
     return polkit.Result.YES;
}
```

Запись в журнал при остановке ВМ:

```
if (action.id.match("org.libvirt.api.domain.stop") ) {
     polkit.log("action=" + action);
     polkit.log("subject=" + subject);
     return polkit.Result.YES;
}
```

3.10. Регистрация событий

3.10.1. Регистрация событий libvirt

Настройка регистрации событий в libvirt, осуществляется в файле `/etc/libvirt/libvirtd.conf`. Логи сохраняются в каталоге `/var/log/libvirt`.

Функция журналирования в libvirt основана на трех ключевых понятиях:

- сообщения журнала;
- фильтры;
- формат ввода.

Сообщения журнала – это информация, полученная во время работы libvirt. Каждое сообщение включает в себя уровень приоритета (отладочное сообщение – 1, информационное – 2, предупреждение – 3, ошибка – 4). По умолчанию, `log_level=1`, т. е. журналируются все сообщения.

Фильтры – это набор шаблонов и для записи сообщений в журнал. Если категория сообщения совпадает с фильтром, приоритет сообщения сравнивается с приоритетом фильтра, если она ниже сообщение отбрасывается, иначе сообщение записывается в журнал. Если сообщение не соответствует ни одному фильтру, то применяется общий уровень. Это позволяет, например, захватить все отладочные сообщения для Qemu, а для остальных, только сообщения об ошибках.

Формат для фильтра:

```
x:name    (log message only)
x:+name (log message + stack trace)
```

где:

- `name` – строка, которая сравнивается с заданной категорией, например, `remote`, `qemu`, или `util.json`;
- `+` – записывать каждое сообщение с данным именем;
- `x` – минимальный уровень ошибки (1, 2, 3, 4).

Пример фильтра:

```
Log_filtrers="3:remote 4:event"
```

Как только сообщение прошло через фильтрацию набора выходных данных, формат вывода определяет, куда отправить сообщение. Формат вывода также может фильтровать на основе приоритета, например, он может быть полезен для вывода всех сообщений в файл отладки.

Формат вывода может быть:

- `x:stderr` – вывод в STDERR;
- `x:syslog:name` – использовать системный журнал для вывода и использовать данное имя в качестве идентификатора;
- `x:file:file_path` – вывод в файл, с соответствующим `filepath`;
- `x:journal` – вывод в `systemd` журнал.

Пример:

```
Log_outputs="3:syslog:libvirtd 1:file:/tmp/libvirt.log"
```

Журналы работы ВМ под KVM хранятся в `/var/log/libvirt/qemu/`. В этом каталоге libvirt хранит журнал для каждой ВМ. Например, для машины с названием `alt-server` журнал будет находиться по адресу:

```
/var/log/libvirt/qemu/alt-server.log
```

3.10.2. Регистрация событий запуска (завершения) работы компонентов виртуальной инфраструктуры

В каталоге `/var/log/libvirt/qemu/` KVM хранит журнал для каждой ВМ. Например, для машины с названием `alt1` журнал будет находиться по адресу `/var/log/libvirt/qemu/alt1.log`.

В этот журнал попадают записи вида:

```
qemu: terminating on signal 15 from pid 118813
2016-12-16 14:39:41.045+0000: shutting down
qemu: terminating on signal 15 from pid 2056
2016-12-19 14:01:55.917+0000: shutting down
2016-12-19 14:02:09.841+0000: starting up libvirt version: 1.3.2,
package: alt1, qemu version: 2.5.0, hostname: vb.office.alt1.ru
```

Можно также вести файл журнала, используя правила polkit. Например, делать запись в журнал при старте ВМ:

```
if (action.id.match("org.libvirt.api.domain.start") ) {
           polkit.log("action=" + action);
           polkit.log("subject=" + subject);
           return polkit.Result.YES; }
         }
```

Запись в журнал при останове ВМ:

```
if (action.id.match("org.libvirt.api.domain.stop") ) {
           polkit.log("action=" + action);
           polkit.log("subject=" + subject);
           return polkit.Result.YES; }
         }
```

Запись в журнал при изменении ВМ:

```
if (action.id.match("org.libvirt.api.domain.write") ) {
           polkit.log("action=" + action);
           polkit.log("subject=" + subject);
           return polkit.Result.YES; }
         }
```

3.10.3. Регистрация входа (выхода) субъектов доступа в/из гипервизор(а)

Регистрацию событий входа (выхода) субъектов доступа в/из гипервизор(а) можно настроить с помощью правил polkit.

При любом действии с подключениями и хранилищем записывать в журнал:

```
if (action.id.match("org.libvirt.unix. ") ) {
            polkit.log("action=" + action);
            polkit.log("subject=" + subject);
            return polkit.Result.YES; }
          }
```

3.10.4. Регистрация событий входа (выхода) субъектов доступа в/из гостевых ОС

Регистрация событий входа (выхода) субъектов доступа в/из гостевых ОС не производится, так как зависит от ОС, выполняемых в ВМ.

3.10.5. Регистрация изменения прав доступа к файлам-образам ВМ

Регистрация событий изменения прав доступа к файлам-образам ВМ можно настроить с помощью `audit`.

Файлы libvirt:

- `/var/lib/libvirt/boot/` – ISO-образы для установки гостевых систем;
- `/var/lib/libvirt/images/` – образы жестких дисков гостевых систем;
- `/etc/libvirt/` – каталог с файлами конфигурации.

Под учетной записью администратора включить контроль над объектом `/var/lib/libvirt/images/`:

```
# auditctl -w /var/lib/libvirt/images/ -p wa
```

В журнале контроля будут фиксироваться записи, свидетельствующие о регистрации факта создания, просмотра и изменения файлов.

4. DOCKER

4.1. Запуск от пользователя

Для запуска docker от пользователя следует выполнить несколько шагов:

1) добавить пользователя в группу `docker`:

```
# usermod ИМЯ_ПОЛЬЗОВАТЕЛЯ -aG docker
```

2) выполнить повторный вход в систему.

Запустить службу:

```
# systemctl enable --now docker
```

5. KUBERNETES

5.1. Подготовка

Нужны несколько машин (nodes), одна из которых будет мастером. Системные требования:

- 2 Гбайт ОЗУ или больше;
- 2 ядра процессора или больше;
- все машины должны быть доступны по сети друг для друга;
- своп должен быть выключен;
- на них должны быть установлены следующие пакеты:

```
# apt-get install docker-ce kubernetes-kubeadm kubernetes-kubelet cri-tools
```

- и запущены сервисы docker, kubelet и kebe-proxy:

```
# systemctl enable --now docker kubelet kube-proxy
```

5.2. Разворачивание кластера

1) На мастере нужно запустить команду для запуска кластера:

```
# kubeadm init --pod-network-cidr=10.244.0.0/16 --ignore-preflight-errors=SystemVerification
```

где:

- `--pod-network-cidr=10.244.0.0/16` – внутренняя (разворачиваемая Kubernetes) сеть, данное значение рекомендуется оставить для правильной работы `Flannel`;
- `--ignore-preflight-errors=SystemVerification` – игнорирование слишком новой версии docker.

В конце вывода будет строка вида:

```
kubeadm join <ip адрес>:<порт> --token <токен> --discovery-token-ca-cert-hash sha256:<хэш>
```

2) Настройка `kubernetes` для работы от пользователя:

- создать каталог `~/.kube`:

```
$ mkdir ~/.kube
```

- от администратора скопировать файл конфигурации:

```
# cp /etc/kubernetes/admin.conf ~<пользователь>/.kube/config
```

- изменить владельца файла конфигурации:

```
# chown <пользователь>: ~<пользователь>/.kube/config
```

3) Подключить к мастеру все остальные ноды:

```
# kubeadm join <ip адрес>:<порт> --token <токен> --discovery-token-ca-cert-hash sha256:<хэш> --ignore-preflight-errors=SystemVerification
```

Проверить наличие нод можно так:

```
$ kubectl get nodes -o wide
```

Пример вывода:

```
NAME     STATUS    ROLES     AGE      VERSION   INTERNAL-IP
EXTERNAL-IP   OS-IMAGE     KERNEL-VERSION       CONTAINER-
RUNTIME
docker1   Ready     <none>    4h       v1.11.2   10.10.3.23
<none>       ALT Regular   4.17.14-un-def-alt1   docker://Unknown
docker2   Ready     <none>    4h       v1.11.2   10.10.3.120
<none>       ALT Regular   4.17.14-un-def-alt1   docker://Unknown
docker3   Ready     <none>    4h       v1.11.2   10.10.3.157
<none>       ALT Regular   4.17.14-un-def-alt1   docker://Unknown
k8s      Ready     master    4h       v1.11.2   10.10.3.227
<none>       ALT Regular   4.17.14-un-def-alt1   docker://Unknown
```

4) Далее следует развернуть сеть. Для этого можно запустить команду:

```
$ kubectl apply –f
```

Проверить работу – выполнить команду:

```
$ kubectl get pods --namespace kube-system
```

Пример корректного вывода:

```
NAME                              READY   STATUS    RESTARTS   AGE
coredns-78fcdf6894-6trk7          1/1     Running   0          2h
coredns-78fcdf6894-nwt5l          1/1     Running   0          2h
etcd-k8s                          1/1     Running   0          2h
kube-apiserver-k8s                1/1     Running   0          2h
kube-controller-manager-k8s       1/1     Running   0          2h
kube-flannel-ds-894bt             1/1     Running   0          2h
kube-flannel-ds-kbngw             1/1     Running   0          2h
kube-flannel-ds-n7h45             1/1     Running   0          2h
kube-flannel-ds-tz2rc             1/1     Running   0          2h
kube-proxy-6f4lm                  1/1     Running   0          2h
kube-proxy-f92js                  1/1     Running   0          2h
kube-proxy-qkh54                  1/1     Running   0          2h
kube-proxy-szvlt                  1/1     Running   0          2h
kube-scheduler-k8s                1/1     Running   0          2h
```

Следует обратить внимание, что `coredns` находятся в состоянии `Running`. Количество `kube-flannel` и `kube-proxy` зависит от общего числа нод (в данном случае их четыре).

5.3. Тестовый запуск nginx

1) Создать Deployment:

```
$ kubectl apply -f
```

2) Затем создать сервис, с помощью которого можно получить доступ к приложению из внешней сети.

Сохраните в файл `nginx-service.yaml` следующую конфигурацию:

```
apiVersion: v1
kind: Service
metadata:
   name: nginx
   labels:
        app: nginx
spec:
   type: NodePort
```

```
  ports:
  - port: 80
       targetPort: 80
  selector:
        app: nginx
```

3) Запустите новый сервис:

```
$ kubectl apply -f nginx-service.yaml
```

4) Чтобы узнать его порт выполните команду:

```
$ kubectl get svc nginx
```

Пример вывода:

```
NAME     TYPE       CLUSTER-IP     PORT(S)   EXTERNAL-IP   AGE
nginx   NodePort  10.108.199.141   <none>   80:32336/TCP   4h
```

5) Проверьте работу:

```
$ curl <IP-адрес>:<порт>
```

где:

- `IP-адрес` – это адрес любой из нод;
- `порт` – это порт сервиса, полученный с помощью предыдущей команды.

Если использовать данные из примеров, то возможная команда: `curl 10.10.3.120:32336`.

6. УДАЛЕННОЕ ПОДКЛЮЧЕНИЕ К ВМ

6.1. VNC подключение к ВМ

К консоли или рабочему столу ВМ KVM можно удаленно подключиться по протоколу VNC. Настройка удаленного доступа к ВМ KVM по протоколу VNC выполняется с помощью менеджера ВМ (virt-manager).

В virt-manager необходимо открыть панель свойств ВМ и раздел «Дисплей Spice». В поле тип выберите VNC-сервер (рис. 30).

VNC-сервер, по умолчанию, обслуживает только локальные VNC-запросы (Localhost only). При таких параметрах удаленный доступ к ВМ будет запрещен.

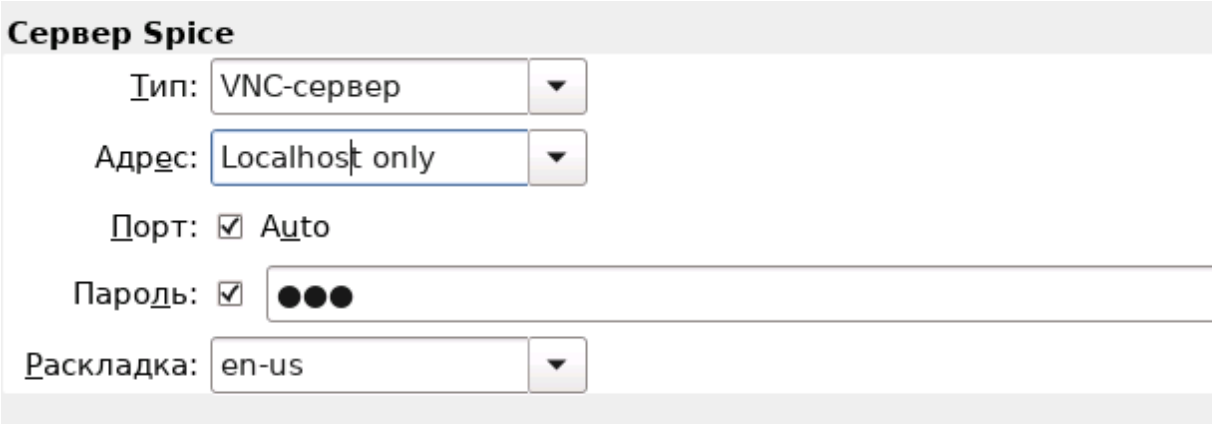

Рис. 30 – VNC сервер

Для удаленного подключения к ВМ KVM по VNC-протоколу VNC-сервер хост системы должен обслуживать запросы с общедоступных сетевых интерфейсов. Для разрешения удаленного доступа на месте адреса необходимо выбрать «All interfaces» (рис. 31).

Если на хосте несколько ВМ, к каждой из них можно будет подключиться через один IP-адрес и разные порты. Порт доступа к ВМ может быть назначен вручную или автоматически. Удаленный доступ к ВМ можно защитить паролем.

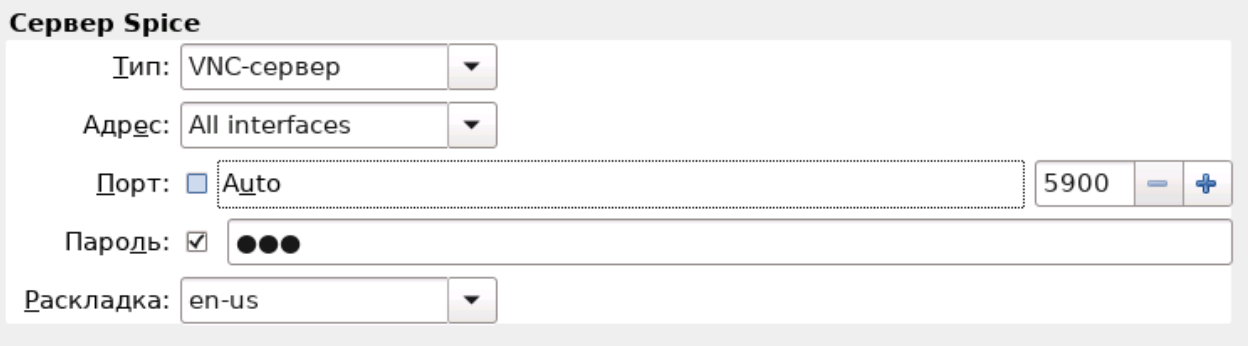

Рис. 31 – Разрешение удаленного доступа

6.1.1. Подключение VNC-клиента к удаленному компьютеру

Для подключения VNC-клиента к удаленному компьютеру требуется указать его IP-адрес или DNS-имя, и номер дисплея (по умолчанию, 0) или номер TCP-порта (по умолчанию, `5900`). Если VNC-сервер требует авторизации, то при подключении к нему VNC-клиент запросит пароль. Пароль доступа к VNC-серверу не связан с каким-либо аккаунтом (учетной записью пользователя) на удаленном компьютере, а служит только для ограничения доступа к дисплею VNC-сервера.

После установки соединения и открытия экрана, в зависимости от настроек VNC-сервера может потребоваться авторизация пользователя на виртуальном сервере или может быть открыта уже запущенная рабочая сессия какого-либо пользователя.

Так как на компьютере одновременно могут работать несколько VNC-серверов, для их разделения используют параметр номер дисплея. Например, один VNC-сервер может быть запущен на дисплее `:0`, другой – на дисплее `:1`. Каждому номеру дисплея соответствует номер TCP-порта, на котором VNC-сервер принимает соединения. Номер порта для дисплея получается прибавлением номера дисплея к базовому номеру порта – `5900`. Дисплею `:0` соответствует TCP-порт `5900`, дисплею `:1` – порт `5901`.

6.1.2. Отключение VNC-клиента от удаленного компьютера

При закрытии окна VNC-клиента или после выхода из окружения средствами рабочего стола, в зависимости от настроек VNC-сервера, рабочая сессия пользователя может закрыться с остановкой всех используемых программ, или продолжать работу и быть доступной снова при повторном подключении к VNC-серверу.

6.2. SPICE подключение к ВМ

К консоли или рабочему столу ВМ KVM, можно удаленно подключиться по протоколу SPICE.

SPICE (Simple Protocol for Independent Computing Environments) – открытый протокол удаленного доступа к компьютеру или ВМ.

Протокол SPICE состоит из следующих элементов:

- SPICE-сервер – обычно QEMU/KVM гипервизор;
- SPICE-клиент – клиент удаленного доступа;
- SPICE-агент – гостевое дополнение, расширяющее интеграцию гостя и хоста;
- SPICE-сервер – графическая подсистема гипервизора.

Чтобы задействовать протокол SPICE в QEMU/KVM необходимо подключить протокол удаленного доступа к графической подсистеме SPICE.

Настройка удаленного доступа к ВМ KVM по протоколу SPICE выполняется с помощью менеджера ВМ (`virt-manager`).

В virt-manager нужно открыть панель свойств ВМ и раздел «Дисплей Spice». В поле тип необходимо выбрать «Сервер spice» (рис. 32).

SPICE-сервер, по умолчанию, обслуживает только локальные SPICE-запросы (Localhost only). При таких параметрах удаленный доступ к ВМ будет запрещен.

Для удаленного подключения к ВМ KVM по SPICE-протоколу SPICE-сервер хост системы должен обслуживать запросы с общедоступных сетевых интерфейсов. Для разрешения удаленного доступа на месте адреса выберите «All interfaces».

Порт доступа к ВМ может быть назначен вручную или автоматически. Удаленный доступ к ВМ можно защитить паролем.

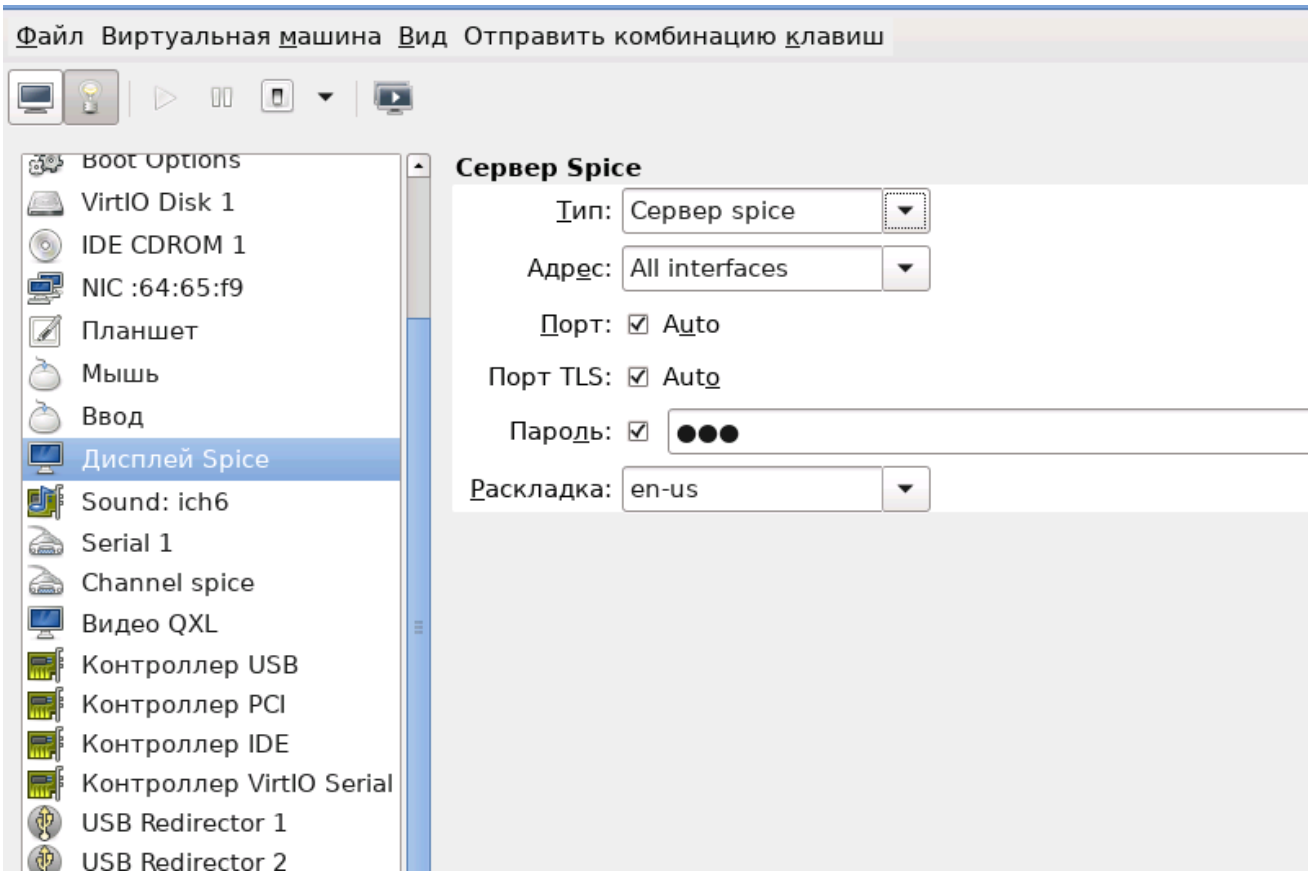

Рис. 32 – Раздел «Дисплей Spice»

SPICE-агент – дополнительный механизм связи гостя с хостом. Предоставляет следующие возможности:

- передача доступа к мыши гостю;
- совместная работа с буфером обмена.

Также необходимо в конфигурацию добавить SPICE-канал (`spicevmc`) для обеспечения связи между хостом и гостем. Это можно сделать в `virt-manager`.

Сохраните настройки, нажав на кнопку «Завершить».

6.3. Проброс USB-устройств в ВМ через SPICE

Протокол удаленного доступа SPICE позволяет не только передавать данные устройств ввода-вывода, но и передавать по сети трафик USB-устройств – пробрасывать USB-устройства клиента без использования дополнительных программ USB-серверов, таких как `usbip`.

Проброс устройств может использоваться для получения и передачи данных на удаленные устройства из ВМ, например, на принтеры, USB-ключи, FLASH-накопители и другие низкоскоростные устройства.

Для настройки возможности проброса устройства необходимо разместить на сервере ВМ KVM под управлением SPICE с поддержкой USB redirect. В самой ВМ установка гостевых дополнений не требуется. На ВМ клиента необходимо установить Linux и SPICE и получить права администратора.

Далее с помощью SPICE требуется осуществить подключение и пробросить USB-устройство через меню «Input» → «Select USB devices for redirection» либо с помощью комбинации клавиш <Shift>+<F10>.

Отключение устройств осуществляется аналогичным способом.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

ВМ – виртуальная машина;

ОЗУ – оперативное запоминающее устройство;

ОС – операционная система;

ПИ – программное изделие;

ПО – программное обеспечение;

ПЭВМ – персональная электронно-вычислительная машина;

ЦПУ – центральное процессорное устройство.

Лист регистрации изменений										
Номера листов (страниц)					*Всего листов (страниц) в докум.*	*№ документа*	*Входящий № сопроводительного докум. и дата*	*Подп.*	*Дата*
Изм.	*измененных*	*замененных*	*новых*	*анулированных*					