Лабораторная работа №1
«Администрирование виртуальных машин и журналирование»

Цель работы: получить базовые навыки работы с виртуальными машинами.

Продолжительность работы: 4 часа

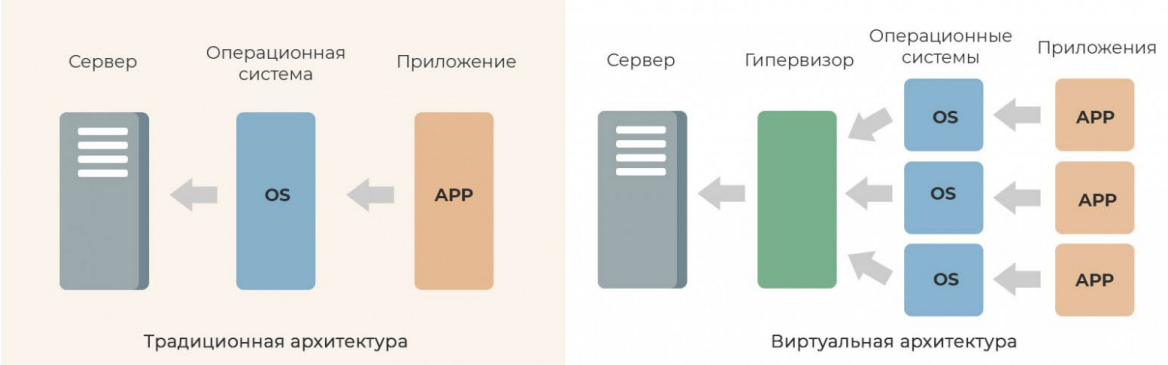
1. Теоретическая часть

1.1. Виртуальная машина

Каждому администратору сети приходится иметь дело с настройкой виртуальных машин. Они необходимы для разработки и тестирования приложений, а также хранения данных. Преимуществами виртуальных машин можно назвать гибкость в выборе операционной системы и возможность дублирования рабочего пространства.

Виртуальная машина (VM, от англ. virtual machine) — программная или аппаратная система, эмулирующая аппаратное обеспечение компьютера и исполняющая программы для guest-платформы (guest — гостевая платформа) на host-платформе (host — хост-платформа, платформа-хозяин).

На одном хост-компьютере может быть множество гостевых ВМ. Хотя виртуальная машина создается с помощью ПО, она использует физические ресурсы хост-машины, такие как ЦП, ОЗУ и место в хранилище на жёстком диске. На своём хост-компьютере можно настроить столько виртуальных машин, сколько нужно, но придётся разделить физические аппаратные ресурсы между ними всеми. Однако большинство ВМ будут работать медленнее, чем физический компьютер, просто из-за дополнительных уровней абстракции, которые они должны пройти для выполнения функции.



Видов виртуализации существует несколько:

Программная виртуализация – вид виртуализации, который использует различные библиотеки ОС, транслируя вызовы виртуальной машины в вызовы ОС. Например, DOSBox, Virtualbox, VirtualPC.

Аппаратная виртуализация – вид виртуализации, который предусматривает специализированную инструкцию аппаратной части, а конкретно инструкций процессора. Позволяет исполнять запросы в обход гостевой ОС прямо на аппаратном обеспечении.

Программная виртуализация обеспечивается гипервизором.

Гипервизор (англ. Hypervisor) или монитор виртуальных машин — программа или аппаратная схема, обеспечивающая одновременное, параллельное выполнение нескольких операционных систем на одном и том же хост-компьютере.

Гипервизор также обеспечивает изоляцию операционных систем друг от друга, защиту и безопасность, разделение ресурсов между различными запущенными ОС и управление ресурсами.

1.2. Виртуализация QEMU/KVM в Astra Linux

Для исполнения прямых аппаратных запросов в ОС должна иметься библиотека, которая направляла бы эти запросы аппаратной части напрямую. На платформах базы Linux долгое время никакой встроенной системы

виртуализации (встроенного гипервизора), просто не существовало. Каждый производитель ПО для виртуализации, который поддерживало технологию аппаратной виртуализации, вынуждены были создавать собственные модули для ядра Linux.

Однако со временем был разработан свободный базовый гипервизор **KVM** или **Kernel-based Virtual Machine**, который представляет из себя загружаемый модуль ядра Linux и предназначен для обеспечения виртуализации на платформ **x86** с поддержкой одной из технологий аппаратной виртуализации — **Intel VT** либо **AMD SVM**.

Сам по себе KVM не выполняет эмуляции. Вместо этого сторонняя программа, работающая в пространстве пользователя, использует интерфейс **/dev/kvm** для настройки адресного пространства гостя виртуальной машины. Например, свободная программа **QEMU** (Quick Emulator) использует KVM для эмуляции аппаратного обеспечения различных платформ.

В Astra Linux по умолчанию используется менеджер виртуальных машин **virt-manager**, представляющий собой пользовательский интерфейс для управления виртуальными машинами на рабочем столе с помощью API **libvirt**. В нём представлена сводная информация о запущенных доменах, их текущей производительности и статистике использования ресурсов. Мастера позволяют создавать новые домены, а также настраивать распределение ресурсов домена и виртуальное оборудование.

1.2.1. Управление виртуальными машинами с помощью утилиты virsh

virsh — утилита, из состава библиотеки libvirt, для командной строки Linux, предназначенная для управления виртуальными машинами и гипервизорами KVM и Xen. Для подключения к гипервизору необходимо указать параметр --connect qemu:///system. Для того, чтобы не указывать каждый раз этот параметр экспортируйте переменную:

```
export LIBVIRT_DEFAULT_URI=qemu:///system
```

Основные команды:

```
virsh start <VM name> – запустить виртуальную машину;
virsh suspend <VM name> – остановить виртуальную машину;
virsh resume <VM name> – возобновить работу виртуальной машины;
virsh shutdown <VM name> – выключить виртуальную машину;
virsh reboot <VM name> – перегрузить виртуальную машину;
virsh list --all – вывести список всех виртуальных машин;
virsh snapshot-create-as <VM name> <SN name> – создать снимок;
virsh snapshot-list <VM name> – вывести список всех снимков;
virsh snapshot-revert <VM name> <SN name> – восстановиться из снимка;
virsh snapshot-delete <VM name> <SN name> – восстановиться из снимка;
```

1.3. Понятие службы в Linux

В операционной системе GNU/Linux, кроме обычных программ, существуют постоянно работающие утилиты, работающие в фоне и предоставляющие определенные функции пользователям или системе. Такие программы называются **службами**.

Например, для подключения к компьютеру по протоколу **ssh** (Secure Shell) необходимо постоянно запущенное приложение сервера, принимающее и обрабатывающее запросы. В этом качестве выступает служба **sshd**. Для каждого нового соединения создаётся её экземпляр, который выполняет обмен ключами, шифрование, аутентификацию, выполнение команд и обмен данными.

Для работы со службами используется утилита **systemctl**, имеющий следующий синтаксис:

```
$ systemctl опции команда служба...
```

Опции служат для настройки поведения запущенной программы. Основные команды перечислим ниже:

- start - запустить службу linux;
- stop - остановить службу linux;
- reload - попросить службу перечитать свою конфигурацию из файловой системы;

- restart - перезапустить службу;
- enable - добавить службу в автозагрузку;
- disable - удалить службу из автозагрузки.

Например, с помощью команды:

```
systemctl start sshd
```

возможно запустить службу для работы с ssh-соединениями.

Для определения месторасположения службы (или любой другой стандартной утилиты) возможно воспользоваться командой whereis

```
whereis ssh
```

1.5. Журналирование

Процесс настройки и поддержки программного обеспечения в ОС Linux является нетривиальной задачей. Для упрощения взаимодействия системного администратора с программным обеспечением утилитам, ядром и приложениями генерируются журнальные данные, которые в дальнейшем возможно обработать и проанализировать.

Журналирование в Linux — это процесс записи событий, сообщений и логов операционной системы и приложений для их последующего анализа, диагностики проблем и аудита безопасности.

Исторически, система UNIX управляла журналами с использованием системы syslog. Это достаточно сложная и громоздкая система, предназначенная для сбора сообщений и их последующей записи в файлы. Из-за её недостатков в дальнейшем многие утилиты разработали свои средства журналирование, что привело еще к большей путанице.

Большинство журналов хранится в директории /var/log/. Рассмотрим некоторые из них.

Название журнала	Основные программы	Содержимое
apt/history.log	apt-get	Сообщения об установке пакетов
auth.log	passwd, polkitd, sshd, su, sudo, useradd, userdel, usermod	Авторизационные сообщения
cron.log	cron	Сведения о работе демона cron
daemon.log	-||-	Сведения о работе различных демонов
dmesg	Ядро	Сообщения ядра ОС
dpkg.log	dpkg	Журнал управления пакетом
kern.log	Ядро	Все сообщения от ядра ОС
mail.log	Почтовые программы (postfix)	Все сообщения, связанные с электронной почтой
syslog	Различные программы	Основной системный журнал

Все перечисленные файлы возможно открыть и прочитать с помощью текстового редактора. Журналы lastlog и wtmp, хранящие в себе сообщения о последней регистрации пользователей в системе хранятся в бинарном виде, поэтому прочитать их возможно с помощью специальных утилит – lastlog и last.

Во многих современных системах, к которым относится в том числе Astra Linux журналирование параллельно ведется системой инициализации systemd. Все события в системе обрабатываются демоном journald, которых сохраняет их в виде бинарных файлов. Для просмотра логов применяется утилита journalctl**.**

Все логи в журнале хранятся в следующем формате:

дата хост источник сообщение

Например,

янв 09 20:55:55 server sshd[1041]: Server listening on 0.0.0.0 port 22

- **янв 09 20:55:55** - дата и время события;
- **server** - хост, на котором произошло событие;

- **sshd[1041]**- источник события, обычно это программа или сервис. В данном случае демон ssh, его pid=1041;
- **Server listening on 0.0.0.0 port 22**- само сообщение.

Перечислим основные команды и параметры journalctl. Вызов производится в привилегированном режиме.

Просмотр всего журнала

```
journalctl
```

Просмотр всего журнала с конца

```
journalctl -e
```

Вывод сообщений журнала, отфильтрованные по коду важности. journalctl выводит все сообщения с этим кодом и выше

```
journalctl -p <код>
```

Для уровней важности, приняты следующие обозначения:

0: emergency (неработоспособность системы)

1: alerts (предупреждения, требующие немедленного вмешательства)

2: critical (критическое состояние)

3: errors (ошибки)

4: warning (предупреждения)

5: notice (уведомления)

6: info (информационные сообщения)

7: debug (отладочные сообщения)

Просмотр журнала во временном промежутке

```
journalctl --since "2020-12-17" --until "2020-12-18 10:00:00
journalctl –since “1 minute ago”
```

Просмотр сообщений ядра

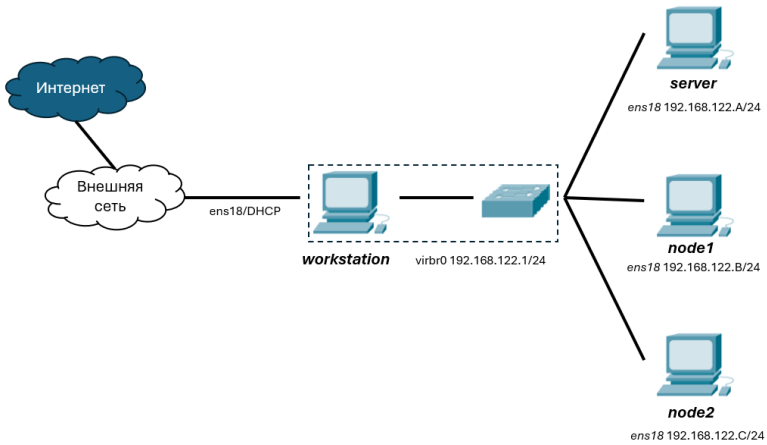
```
journalctl –k
```

Просмотр журналов определенного сервиса

```
journalctl -u NetworkManager.service
```

2. Практическая часть

В рамках лабораторных работ будет предложено работать со стендом, схема которого представлена на рисунке N. Основной машиной является workstation, внутри которой должны быть развернуты три виртуальных машины – server, node1, node2.



В рамках первой лабораторной работы не требуется устанавливать IP адреса или другие сетевые настройки машин. Это будет сделано в рамках второй лабораторной работы.

1. Основная настройка

Подключитесь к виртуальной машине. Откройте вкладку терминала. Перейдите в раздел «Настройка -> Настроить профиль». В открывшейся панели вы можете настроить терминал под свой вкус.

Рекомендуем указать следующие параметры:

- Вкладки -> Удаленный сеанс. Изменить значение на %h. Это позволит вывести вместо номера вкладки имя подключаемой по ssh машины.
- Мышь -> Установить галочку «Копировать при выделении».

2. Работа с виртуальными машинами

Работу с виртуальными машинами возможно осуществлять с использованием командной строки с помощью утилиты virsh или с использованием графического приложения «Менеджер виртуальных машин».

Для работы необходимо наличие трех виртуальных машин – server, node1, node2. Создайте их путем клонирования существующей машины alse18. Для этого войдите в приложение «Менеджер виртуальных машин» и щелкните правой кнопкой мыши по строке alse18. В выпадающем меню выберете пункт «Клонировать». Название машины должно совпадать с названием со схемы (server, node1, node2).

Перед выполнением всех заданий создайте снимок каждой из машин. Дайте им осмысленные названия. После окончания каждой из работ создавайте снимки данных. По умолчанию, если об этом не сказано в тексте, задания выполняются на ВМ server.

Запустите машины. После этапа загрузки операционной системы появится окно для ввода пароля пользователя **adminstd**. Введите пароль по умолчанию **qwerty22**. Используя эмулятор терминала Fly, выполните следующие задания.

Задание 1.

2.1. Создайте на каждой из машин пользователя. Имя пользователя должно иметь формат: фамилияИО. Например, ivanovii. Все дальнейшие действия должны производиться из-под этого пользователя.

2.2. Добавьте пользователя в группы sudo и во все группы, в которые входит пользователь adminstd. Выдайте созданному пользователю максимальную метку целостности (63).

2.3. Войдите в систему из-под созданного пользователя и на сервере установите утилиту, позволяющую работать по протоколу ftp. (Подсказка, в качестве ключевых слов поиска используйте lightweight, efficient, ‘ftp server’)

Задание 2.

1. С помощью команды journalctl найдите сообщения, поступившие в журнал службы systemd-journald с 12 до 17 часов вчерашнего дня.
2. Определите по значениям логов, сколько пользователей было создано и удалено в системе. Проанализируйте, какой из пользователей не смогу войти в свою учетную запись при вводе пароля в окне логина. Создайте еще одного нового пользователя, попробуйте авторизироваться в системе, допустив ошибку в пароле. Удалите созданного пользователя. Убедитесь, что все действия были отображены в системном журнале.
3. Определите, какая утилита чаще всего писала в файл syslog

Задание 3.

1. Выключите машины и создайте снимок каждой из них. Дайте им осмысленное название.
2. Подключитесь к машине node1 и выполните команду sudo rm -rf /. К какому результату это привело? Подключитесь к машине node2 и выполните команду :(){ :|:& };: К какому результату это привело? Объясните результат выполнения команды (см. https://www.cyberciti.biz/faq/understanding-bash-fork-bomb/).
3. Восстановите машины, используя созданные ранее снимки.

Контрольные вопросы

1. На какой странице man возможно прочитать про системные вызовы?
2. В каком файле хранятся сообщения от ядра ОС?
3. В чем отличие в хранении логов syslog и journalctl?