**LINUX EXERCISE (LAB 05)**

Exercise 1. Install and run any operating system using Virt Manager (Report step by step of your progress)

\*Solve:

sudo apt update

sudo apt upgrade

Cài qemu: sudo apt install qemu-kvm libvirt-daemon-system libvirt-clients bridge-utils virtinst virt-viewer

sudo apt install dialog nếu lệnh trên bị lỗi

Virt Manager cần libvirtd service to manage virtualization. Start libvirtd service: sudo systemctl start libvirtd

Để start nó cần systemd chạy mà trong wsl2, systemd k được chạy tự động, nếu nó chạy sẽ có hàng đống task background chạy khi bật lên. Để fix systemd:

sudo -e /etc/wsl.conf

Thêm:

[boot]

systemd=true

Thoát khỏi nano với: ctrl + X

Thoát khỏi wsl và chạy: wsl --shutdown

Khởi động lại wsl và start được libvirtd với: sudo systemctl start libvirtd

Có thể check systemd đang chạy với: sudo systemctl status

sudo apt install virt-manager

Chạy virt manager với: virt-manager

Download iso của bất cứ hệ điều hành nào về

Virt manager sẽ connect với qemu đang chạy, từ đó, ta chạy các hệ điều hành trên nó:

Create new virtual machine -> trỏ tới đường dẫn file ISO tải về

Try and install ubuntu và chạy được ubuntu trên qemu

upgrade virt manaer: sudo apt update && sudo apt upgrade

uninstall virt manager: sudo apt autoremove --purge virt-manager

=> Kiến trúc ta đang dùng với wsl rất ảo:

Đầu tien là Window chạy wsl2 dạng hypervisor là 1 dạng ảo hóa kiểu:

App -> OS -> hypervisor layer -> OS host thực sự -> hardware

Sau đó ta cài vào ubuntu: là OS muốn chạy trong môi trường ảo

Bên trong môi trường ảo này ta lại cài QEMU: là 1 emulator cho phép chạy các hệ điều hành khác trên nó và tương tác với commandline. Nó như 1 ứng dụng bth, tức là phần app bên trên:

QEMU -> Ubuntu -> hypervisor layer -> OS host thực sự -> hardware

Nhưng do emulator chậm, ta dùng thêm KVM. 1 công nghệ ảo hóa ốp thẳng vào nhân linux:

QEMU -> Ubuntu(kernel được integrated KVM module) -> hypervisor layer -> OS host thực sự -> hardware

KVM giúp hệ điều hành nó integrated vào chạy được nhiều máy ảo vật lý và nó giúp quản lý đống máy ảo đó:

QEMU chạy android -> |

QEMU chạy Linux -> | Ubuntu(kernel được integrated KVM module) -> ...

QEMU chạy window -> |

Nhưng KVM lúc này ta k tương tác được. Còn QEMU chạy commanline tương tác cũng mệt, nên cài thêm virt manager để dùng KVM quản lý bằng giao diện. Nó chỉ là 1 app thôi:

QEMU -> |

Virt Manager -> | Ubuntu(kernel được integrated KVM module) -> ...

Virt Manager sẽ tìm các QEMU trong hđh hiện tại và connect với nó để thao tác bằng giao diện, như v ta k cần chạy lệnh QEMU thủ công nữa mà nó kiểu:

VirtManager (chạy QEMU) -> Ubuntu(kernel được integrated KVM module) -> hypervisor layer -> OS host thực sự -> hardware

Vài câu hỏi khác:

What does Virt-manager do?

Virt-Manager is an open-source tool that can be used to manage Virtual machines Graphically just like VirtualBox on Linux systems. Popularly it is used for KVM running VMs, however, can manage virtual machines running on other virtualization technologies such as Xen and LXC.

How do I get Virt-manager on Windows?

Virt-Manager is not available for the Windows platform. Only the Virt-Viewer can be used on Windows. However, still, if you want to use the VMM then can go for the WSL method.

Can KVM run on Windows?

No, we cannot install KVM natively on Windows operating system because it is a virtualization module in the Linux kernel that allows the kernel to function as a hypervisor. Hence, Windows operating systems don’t support KVM instead offer their own Hypervisor called Hyper-V.

Is QEMU better than VirtualBox?

No, QEMU is not better than VirtualBox because Oracle Virtualbox is fast, easy to use, and scalable. On the other hand, QEMU is an emulator that is not true virtualization. However, if your system’s CPU doesn’t support Virtualization then QEMU is a good option.

What is the difference between KVM and QEMU?

QEMU is a Quick Emulator that provides a set of different hardware and device models for the OS you want to install. Whereas the KVM is a kernel module for Linux to enable virtualization; this is the hypervisor. Qemu can be used to install Guest operating systems, however, the speed will be slow. That is the reason why developers have combined it with KVM, to run virtual machines at near-native speed.

Kiến trúc:

Quick Emulator: QEMU [Ubuntu VM] | [Window VM] | [Android VM]

Host OS Kernel | KVM integrated

Hardware (CPU, RAM, IO)

Exercise 2. Run a 64-bit Linux operating system in qemu command-line mode using an available image file.

Some instructions for running qemu in command-line mode

* Type *qemu-system-x86\_64 –h* to see all options
* Syntax for running the OS: *qemu-system-x86\_64 OPTIONS IMAGE*
* *IMAGE is the image file of the virtualized operating system*
* *OPTIONS are settings such as RAM size, characteristics, CPU, graphics,...*
* You need the virtual image file of the corresponding operating system (in formats such as raw, qcow2, vmdk,...).
* Example:
* <http://www.nic.funet.fi/pub/mirrors/fedora.redhat.com/pub/fedora/linux/releases/30/Cloud/x86_64/images/Fedora-Cloud-Base-30-1.2.x86_64.qcow2>
* Use wget or curl to download
* Some important options:
* -smp: use multiple cores.
* -boot: boot options.
* -m: memory size.
* -name: set the machine name.
* -drive: describe the disk or image file.
* -nographic: no graphics.
* -net: configure networking for the virtual machine

\*Solve:

Use wget to download the virtual image file of Fedora 30 Cloud image:

wget <https://download.fedoraproject.org/pub/fedora/linux/releases/34/Cloud/x86_64/images/Fedora-Cloud-Base-34-1.2.x86_64.qcow2>

Run command in the downloaded file directory:

qemu-system-x86\_64 -m 2G -smp 2 -boot c -drive file=Fedora-Cloud-Base-34-1.2.x86\_64.qcow2 -nographic

Option:

-m 2G: Sets the memory size to 2GB. You can adjust this value based on your system's resources.

-smp 2: Configures QEMU to use 2 CPU cores. Adjust the value as desired.

-boot c: Specifies that QEMU should boot from the first virtual disk (image file).

-drive file=Fedora-Cloud-Base-30-1.2.x86\_64.qcow2: Defines the virtual disk or image file to be used.

-nographic: Runs QEMU in console mode without graphical output. This option is useful for command-line-only usage.

=> After a long time to finish setup, I can interact with the virtual machine through the command line.



Exercise 3. If you want to use QEMU to create/run an ARM embedded virtual machine, what command should you use? Can you do the same thing with Virt Manager? If so, what do you need to set up?

To use QEMU for creating and running an ARM embedded virtual machine:

```

qemu-system-arm -machine virt -cpu cortex-a53 -m 1G -kernel <path\_to\_kernel\_image> -append "root=/dev/vda2 rw console=ttyAMA0" -drive file=<path\_to\_rootfs\_image>,format=raw

```

Breakdown option:

- `-machine virt`: Specifies the machine type as "virt" for ARM virtualization.

- `-cpu cortex-a53`: Sets the CPU model to Cortex-A53. You can choose a different CPU model as per your requirements.

- `-m 1G`: Allocates 1GB of memory for the virtual machine. Adjust the value based on your needs.

- `-kernel <path\_to\_kernel\_image>`: Specifies the path to the kernel image file.

- `-append "root=/dev/vda2 rw console=ttyAMA0"`: Provides the kernel command-line parameters. Adjust them based on your specific requirements.

- `-drive file=<path\_to\_rootfs\_image>,format=raw`: Specifies the path to the root filesystem image file and sets the format to raw.

Make sure to replace `<path\_to\_kernel\_image>` with the actual path to the ARM kernel image file, and `<path\_to\_rootfs\_image>` with the path to the root filesystem image file.

Regarding Virt Manager, it can also be used to create and manage virtual machines for ARM architectures. However, setting up Virt Manager for ARM virtualization requires additional configuration steps. Here's what you need to set up:

1. \*\*Install required packages\*\*: Ensure that you have the necessary packages installed for ARM virtualization. For example, on Ubuntu, you can install the required packages by running:

```

sudo apt-get install qemu-kvm libvirt-daemon-system libvirt-clients bridge-utils virt-manager

```

2. \*\*Configure libvirt\*\*: Edit the `/etc/libvirt/qemu.conf` file and uncomment or add the following line to allow non-root users to run virtual machines:

```

user = "your\_username"

```

3. \*\*Configure Virt Manager\*\*: Launch Virt Manager and go to "Edit" > "Connection Details". In the "Virtualization" tab, ensure that "Enable storage and network options for virtualization" is checked. This step allows Virt Manager to handle ARM-based virtual machines.

Once you have completed the setup, you can launch Virt Manager and create ARM virtual machines using its graphical interface. It provides a user-friendly way to configure the virtual machine settings, including the CPU model, memory, kernel, root filesystem, and other options.

Tùy vào phiên bản linux mà có thể sẽ khác