CSIE 3015 Homework 1

In this homework, we will be running various kinds of virtual machines, compare their performance, and have some discussions.

TA information

The TA for this homework is 徐翊凌. If you have any questions regarding the homework assignment, you can contact 徐翊凌 at r11944042@csie.ntu.edu.tw. Yi-Lin's office is located in R404.

Office hour

We will have special office hours for this homework. If you have any questions about this homework, feel free to visit **R404** on **2023/3/14** and **2023/3/21, 10 AM to 11AM**.

1. Host VM setup

You should install Ubuntu on a VM (on VMware Player, VirtualBox or Parallels Desktop), which you are allowed to install any software packages as you wished. The tutorial provided as follows is based on Ubuntu 20.04 LTS. Please make sure to have at least 20GB free storage in your Ubuntu environment.

Now, inside the host VM, we will be running various kinds of virtual machines, including system virtual machine (which we referred as "guest VM" here), Container, Firecracker, and gVisor.

2. Host VM Environment setup

a. Installing QEMU

First, we will install some dependencies for QEMU.

```
$ sudo apt update
$ sudo apt install git build-essential libglib2.0-dev libfdt-dev
libpixman-1-dev zlib1g-dev ninja-build
```

Then, clone QEMU from the repo and checkout to version v6.1.0.

```
$ git clone https://gitlab.com/qemu-project/qemu.git
$ cd qemu/
$ git checkout tags/v6.1.0
```

Configure, compile, install, and leave the qemu directory.

```
$ ./configure --target-list=x86_64-softmmu --disable-werror
$ make -j4
$ sudo make install
$ cd ..
```

b. Getting the image for our guest VM

We will be using Debian cloud image as the image for our guest VM. Download the image and resize to the disk size we need.

```
$ wget -O guest.img
https://cloud.debian.org/images/cloud/bullseye/latest/debian-11-nocloud-
amd64.qcow2
$ qemu-img resize guest.img 10g
```

3. Running the geust VM

Now that the environment has been set up, we can use QEMU to run the second layer of VM (which we call **guest VM**).

a. Boot guest VM

```
$ qemu-system-x86_64 -nographic -m 2048 -smp 2 \
    -drive if=virtio,file=guest.img,cache=none,format=qcow2 \
    -display none \
    -serial mon:stdio \
    -netdev user,id=net0,hostfwd=tcp::2222-:22 \
    -device virtio-net-pci,netdev=net0,mac=de:ad:be:ef:41:49
```

Type `root` when asked for login. You shouldn't need any password. Now, set the hostname as `${student_id}-guest`. For example, `r12345678-guest`:

```
# hostnamectl set-hostname r12345678-guest
```

`logout` then login again, you should see your shell similar to below:

```
root@12345678-guest:~#
```

c. Shutdown guest VM

```
# shutdown -h now
```

4. Running Container

In **host VM**, follow the guide below to install and run Docker.

a. Install Docker

Follow ***Install using the repository*** from Docker Documentation to install docker.
https://docs.docker.com/engine/install/ubuntu/

b. Run Debian Container in Docker

```
# sudo docker run -it --rm --memory=512m --cpus=2 debian
```

When you see the shell as something similar to `root@11ff84c5c294:/#`, it means you're running Debian Container in Docker. To exit the container, type `exit`.

5. Benchmark guest VM and Container

a. Setup

In the **host VM**, **guest VM** and **Container**, install apache2 and hackbench:

```
# sudo apt update
# sudo apt install apache2 rt-tests
```

b. Running Apachebench

Start Apache server and run ApacheBench:

```
# service apache2 start
# ab -q -n 10000 -c 100 http://localhost/gcc/index.html \
    | grep "Requests per second"
```

c. Running hackbench

Run hackbench:

```
# hackbench
```

6. Submission and grading

a. Record a video of your guest VM and Container running correctly (65%)

Record a video of you booting guest VM and Container. Please make sure the hostname is set to your student id. Save the file as `${STUDENT_ID}.mp4`. For example, `r12345678.mp4`.

b. Questions and answers (30%)

Save the answer of following questions to `${STUDENT_ID}.pdf`. For example, `r12345678.pdf`

- Run ApacheBench and hackbench for three times on **host VM**, **guest VM** and **Container** respectively. Screenshot the results. What might be the reason that there's a performance difference between the three (15%)
- Why virtual machines could provide better security than containers? (15%)

c. **Deadline**

The assignment deadline is **2023/3/24 11:59AM**.

d. **Late submission policy**

If the assignment is submitted late, your assignment grade will be the original grade deducted by the number of days late multiplied by 20, until your assignment becomes a zero.

- late submission grade = max(original grade - n * 20, 0)
    - n: late submission days

e. Bonus (Firecracker and gVisor) (5%)

Firecracker and **gVisor** are two new kinds of virtual machines. **Firecracker** is a Virtual Machine Monitor (VMM) that uses KVM to create lightweight VMs (or microVMs), while **gVisor** implements a user-space kernel written in Go to provide an additional layer of isolation between applications and OS. You can use the links below to learn how to install Firecracker and gVisor. Record the video of you booting them in **host VM**. Save the file as `${STUDENT_ID}_bonus.mp4.` For example, `r12345678_bonus.mp4`.
https://github.com/firecracker-microvm/firecracker/blob/main/docs/getting-started.md
https://gvisor.dev/docs/user_guide/install/

Tips for running Firecracker

- It's recommended to use Firecracker v0.24.6.
- Follow ***Configuring the microVM without sending API requests*** guide to run Firecracker. You still need to download kernel and rootfs.
- Get this config file: https://github.com/firecracker-microvm/firecracker/raw/firecracker-v0.24/tests/framework/vm_config.json (You might need to edit this part: `"path_on_host": "xenial.rootfs.ext4"`)

Tips for running gVisor

- In `/etc/default/grub` of **host VM**, you might need to append `systemd.unified_cgroup_hierarchy=0` to `GRUB_CMDLINE_LINUX=`. Then `update-grub`.
- Use `docker run --runtime=runsc --rm -it debian dmesg` to check you're running gVisor correctly.

7. More tips

- Check which VM you're running! You can check hostname to make sure you're running host, guest, or container.
- Use Ctrl-a x (press Ctrl-a, release, then press x) to force quit QEMU when you're stuck.