

Build a Mobile Application With the Kivy Python Framework

การติดตั้งหรือ Config QEMU มี 2 แบบ

1. แบบที่ลงบน Windows
2. แบบที่ลงบน Linux (Ubuntu)

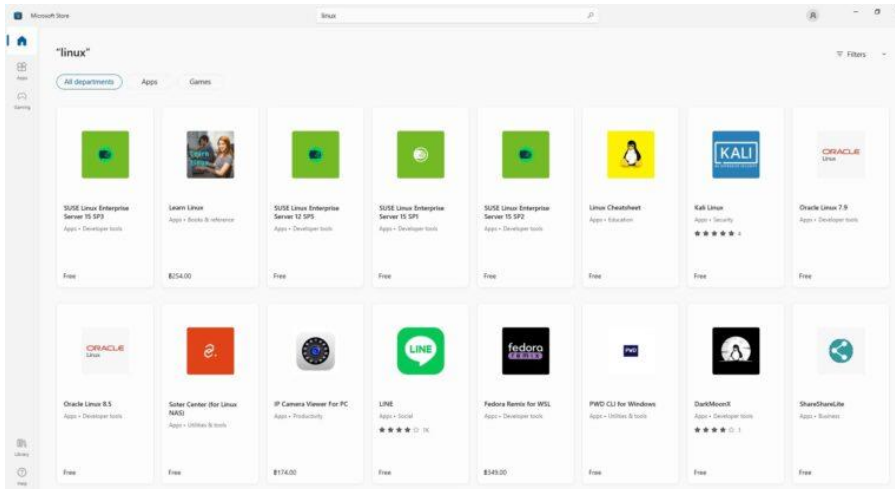
โดยการลง QEMU นั้นจำเป็นที่จะต้อง มี Linux หรือ Ubuntu รองรับเสมอ

1. แบบที่ลงบน Windows

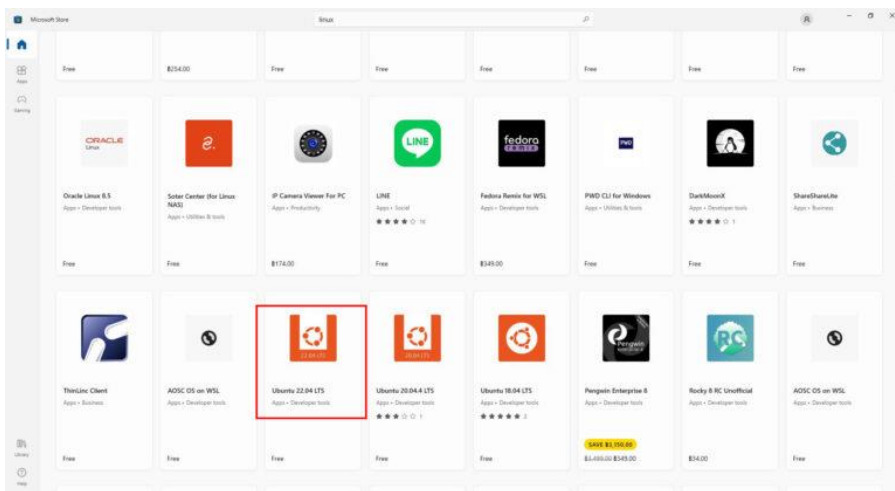
1.1 ขั้นตอนการติดตั้ง WSL (กรณีที่ใช้ใน windows) ***ก่อนลง QEMU ***

****ให้เข้าไปที่ PowerShell (Run as Administrator) ก่อน*****

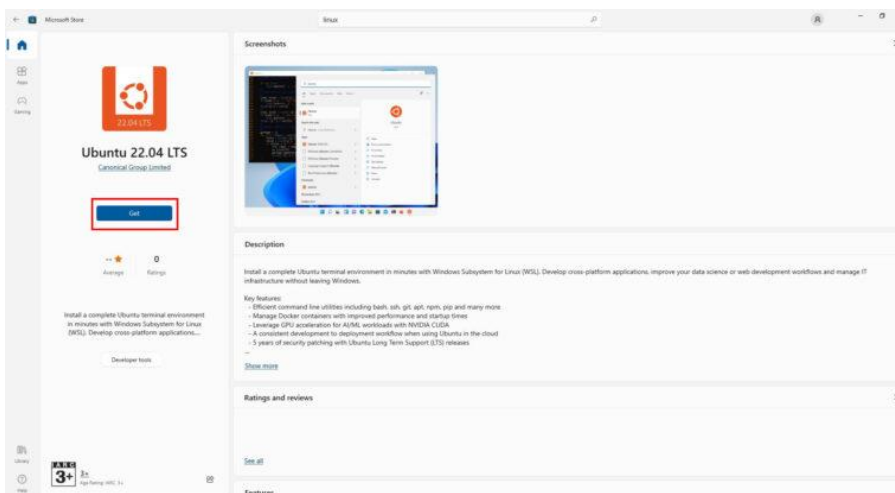
1. `wsl --list --online`
เพื่อเลือก Version ของ Ubuntu ที่เราจะติดตั้ง
2. `wsl --install -d DISTRO-NAME`
ตรง DISTRO-NAME ให้เป็น Version ที่เราเลือกไว้
3. `start-process powershell -verb runas`
เปิด Powershell runs as administrator
4. restart Windows ใหม่
5. `dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart`
6. `dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart`
7. `wsl --set-default-version 2`
8. เข้าไปที่ Microsoft Store โดยเลือกคำสั่ง Start -> Microsoft Store
9. พิมพ์คำว่า linux ในช่องค้นหา แล้วกดปุ่ม Enter หน้าจอ Microsoft Store ก็จะมีหน้าต่างดังรูป



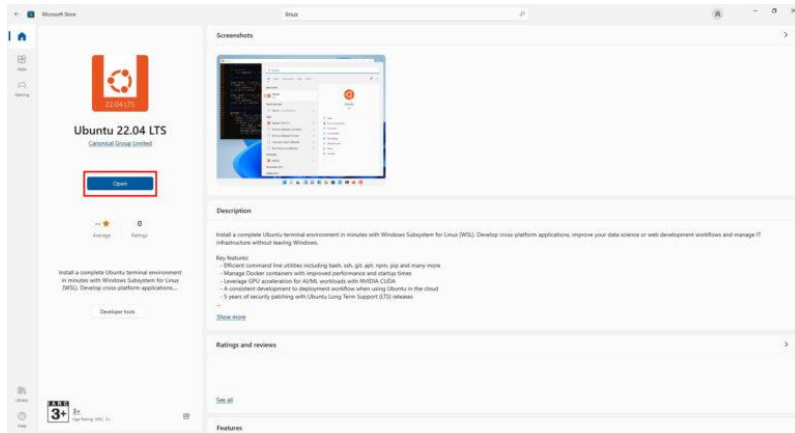
10. เลือก Linux distribution ที่ต้องการติดตั้ง ในที่นี้ เลือกติดตั้ง Ubuntu 22.04 LTS



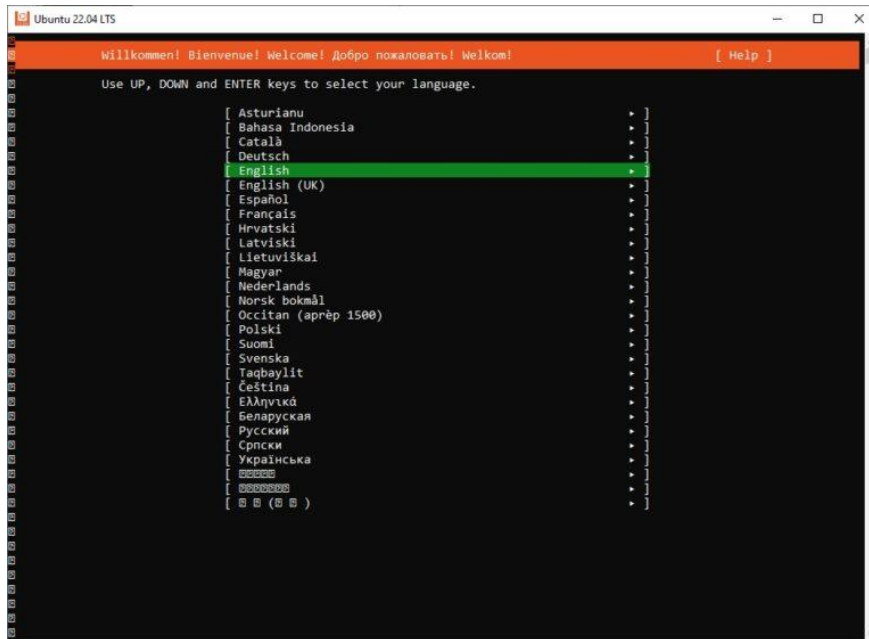
11. ดาวน์โหลด Ubuntu มาติดตั้งบนเครื่องโดยคลิกที่ปุ่ม Get



12. เมื่อดาวน์โหลดเสร็จแล้ว ให้กดปุ่ม Open ดังรูป

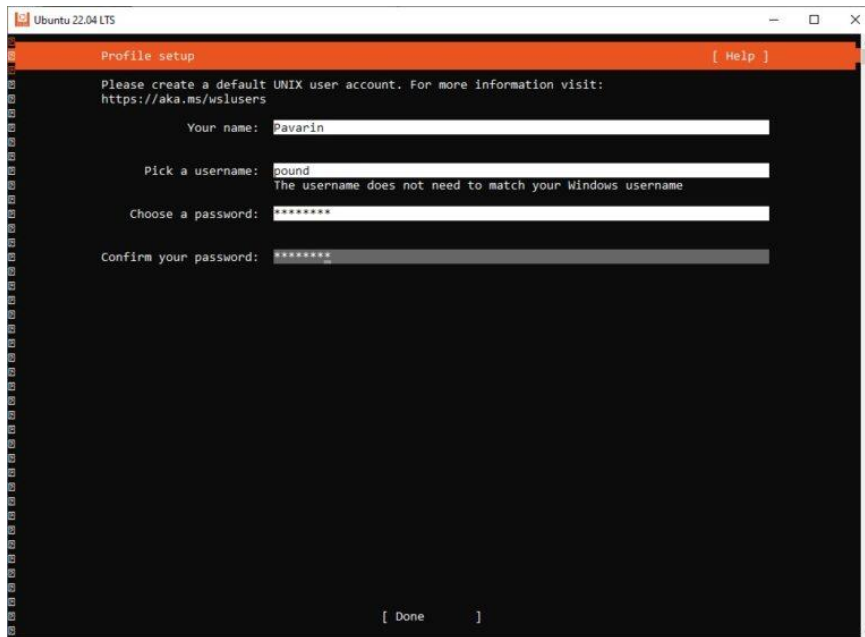


13. หลังจากนั้นก็จะติดตั้ง Ubuntu 22.04 LTS บนเครื่อง เมื่อติดตั้งเสร็จแล้วก็เข้าสู่หน้าจอกำหนดภาษา เริ่มต้นของ Ubuntu ดังรูป



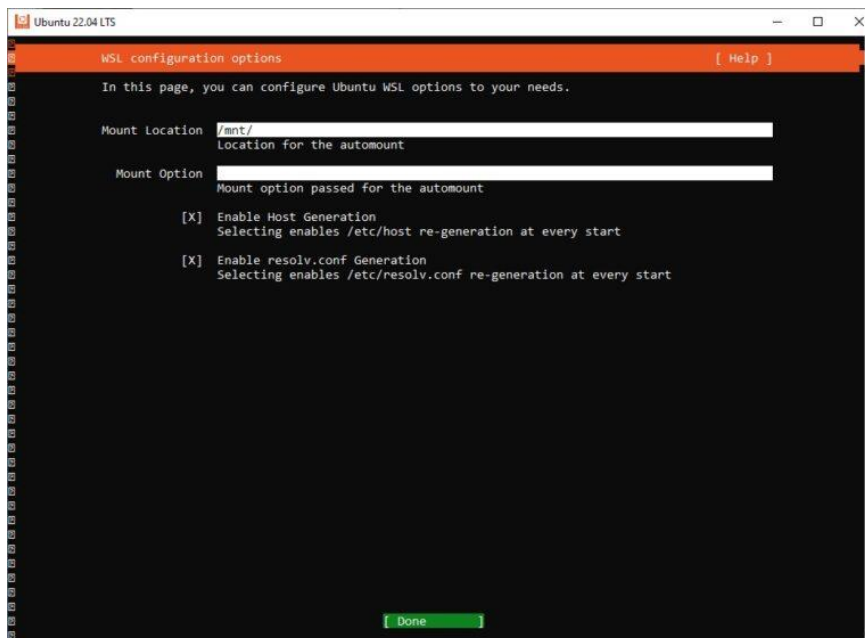
14. ให้กดปุ่มลูกศรเลื่อนลงเพื่อเลือกภาษาเป็น English แล้วกดปุ่ม Enter

15. จากนั้นก็เข้าสู่หน้าจอการสร้างบัญชีผู้ใช้และรหัสผ่าน



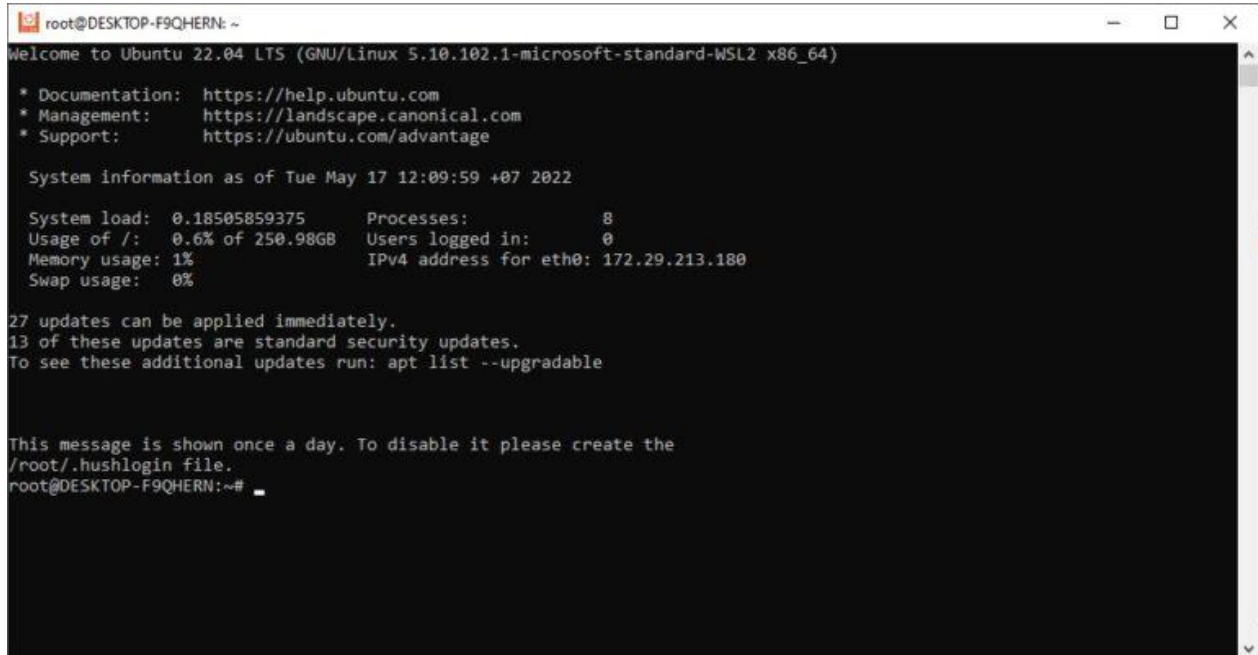
16. เมื่อสร้างเสร็จแล้วให้กดปุ่มเลื่อนลงมาเพื่อไปที่คำว่า Done แล้วกดปุ่ม Enter

17. จากนั้นก็เข้าสู่หน้าจอกำหนดไฟล์เดอร์ที่จะเมากับ WSL



18. ให้กดปุ่ม Enter ก็เป็นอันเสร็จสิ้นการเปิดใช้งาน Windows Subsystem for Linux (WSL) บน Windows

19. ทดสอบได้ด้วยการเปิดโปรแกรม Ubuntu 22.04 โดยเลือกเมนู Start -> Ubuntu 22.04 LTS



```
root@DESKTOP-F9QHERN: ~
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.10.102.1-microsoft-standard-WSL2 x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

System information as of Tue May 17 12:09:59 +07 2022

System load:  0.18505859375   Processes:            8
Usage of /:   0.6% of 250.98GB Users logged in:         0
Memory usage: 1%             IPv4 address for eth0: 172.29.213.180
Swap usage:   0%

27 updates can be applied immediately.
13 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


This message is shown once a day. To disable it please create the
/root/.hushlogin file.
root@DESKTOP-F9QHERN:~#
```

1.2 ขั้นตอนการลง QEMU/KVM on Ubuntu (ในกรณีที่ต้องใช้ ARM64 Arch)

Step 1: Check Virtualization Enabled in Ubuntu

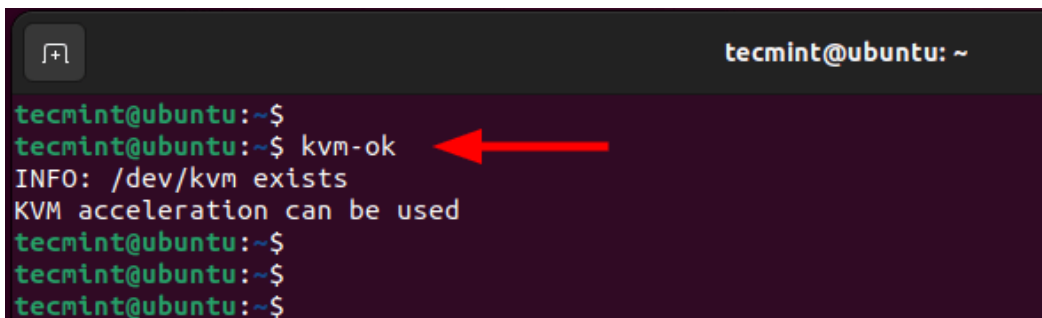
```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

```
$ grep -E --color '(vmx|svm)' /proc/cpuinfo
```



```
tecmint@ubuntu: ~
tecmint@ubuntu:~$
tecmint@ubuntu:~$
tecmint@ubuntu:~$ egrep -c '(vmx|svm)' /proc/cpuinfo
4
tecmint@ubuntu:~$
tecmint@ubuntu:~$
tecmint@ubuntu:~$ grep -E --color '(vmx|svm)' /proc/cpuinfo
flags              : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx f
xsr sse sse2 ss syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon nopl xtopology tsc_reliable nons
top_tsc cpuid tsc_known_freq pni pclmulqdq vmx ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt ts
c_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch cpuid_fault invpcid_sin
gle ssbd ibrs ibpb stibp tpr_shadow vnmi ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 invpci
d_rdtseed adx smap clflushopt xsaveopt xsavec xgetbv1 xsaves arat md_clear flush_l1d arch_capabilities
vmx flags          : vmml invvpid ept_x_only ept_ad tsc_offset vtptr mtf ept vpid unrestricted_guest ple ep
t_node_based_exec
flags              : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx f
xsr sse sse2 ss syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon nopl xtopology tsc_reliable nons
top_tsc cpuid tsc_known_freq pni pclmulqdq vmx ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt ts
c_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch cpuid_fault invpcid_sin
gle ssbd ibrs ibpb stibp tpr_shadow vnmi ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 invpci
d_rdtseed adx smap clflushopt xsaveopt xsavec xgetbv1 xsaves arat md_clear flush_l1d arch_capabilities
vmx flags          : vmml invvpid ept_x_only ept_ad tsc_offset vtptr mtf ept vpid unrestricted_guest ple ep
t_node_based_exec
```

```
$ kvm-ok
```



```
tecmint@ubuntu: ~
tecmint@ubuntu:~$
tecmint@ubuntu:~$ kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
tecmint@ubuntu:~$
tecmint@ubuntu:~$
tecmint@ubuntu:~$
```

Step 2: Install QEMU/KVM on Ubuntu

```
$ sudo apt update
```

```
$ sudo apt install qemu-kvm virt-manager virtinst libvirt-clients bridge-utils libvirt-
daemon-system -y
```

```
$ sudo systemctl enable --now libvirtd
```

```
$ sudo systemctl start libvirtd
```

```
$ sudo systemctl status libvirtd
```

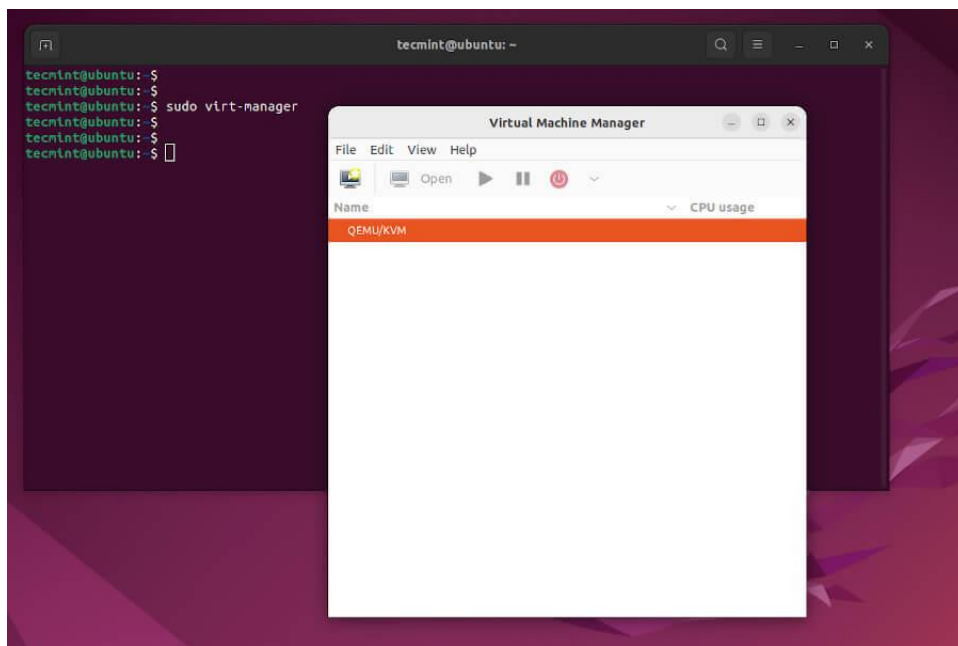
```
tecmint@ubuntu: ~  
tecmint@ubuntu:~$  
tecmint@ubuntu:~$ sudo systemctl enable --now libvirt  
tecmint@ubuntu:~$  
tecmint@ubuntu:~$ sudo systemctl start libvirt  
tecmint@ubuntu:~$  
tecmint@ubuntu:~$ sudo systemctl status libvirt  
● libvirt.service - Virtualization daemon  
   Loaded: loaded (/lib/systemd/system/libvirt.service; enabled; vendor preset: enabled)  
   Active: active (running) since Tue 2022-10-25 11:49:16 EDT; 5min ago  
     TriggeredBy: ● libvirt-ro.socket  
                  ● libvirt.socket  
                  ● libvirt-admin.socket  
     Docs: man:libvirt(8)  
           https://libvirt.org  
    Main PID: 11582 (libvirt)  
      Tasks: 21 (limit: 32768)  
    Memory: 12.8M  
       CPU: 6.416s
```

\$ sudo usermod -aG kvm \$USER

\$ sudo usermod -aG libvirt \$USER

Step 3: Launch Virtual Machine Manager in Ubuntu

\$ sudo virt-manager



ตั้งแต่ Step4 เป็นต้นไปทำตามลิงก์ ด้านล่างได้เลย

ทำตามลิงก์ <https://www.tecmint.com/install-qemu-kvm-ubuntu-create-virtual-machines/>

การตั้งค่าเริ่มต้นของ Virtual Machine Manager

RAM = 2048 CPU = 2

1.3 วิธีการติดตั้งและการใช้งาน Conda (กรณีที่ใช้ใน windows)

1. sudo apt-get update

เพื่ออัปเดต Ubuntu

2. wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh

เพื่อดาวน์โหลด Miniconda ลง Ubuntu

3. sha256sum Miniconda3-latest-Linux-x86_64.sh

ยืนยันไฟล์ที่ดาวน์โหลดมา

4. bash Miniconda3-latest-Linux-x86_64.sh

เพื่อติดตั้ง Miniconda

5. conda create -n myenv python

เพื่อสร้าง env ขึ้นมาใหม่

6. conda deactivate

เพื่อออกจาก env ที่ใช้อยู่

7. conda remove --name ENV_NAME --all

เพื่อ env ที่ไม่ใช่แล้ว

8. conda activate kivy

เพื่อเปิดใช้งาน env ที่เราเลือก

9. python -m pip install --upgrade pip

อัปเดต python

10. sudo apt-get install python3-pip

11. pip list --outdated

12. sudo apt install python3-dev

13. conda install kivy -c conda-forge

ติดตั้ง Kivy Linux

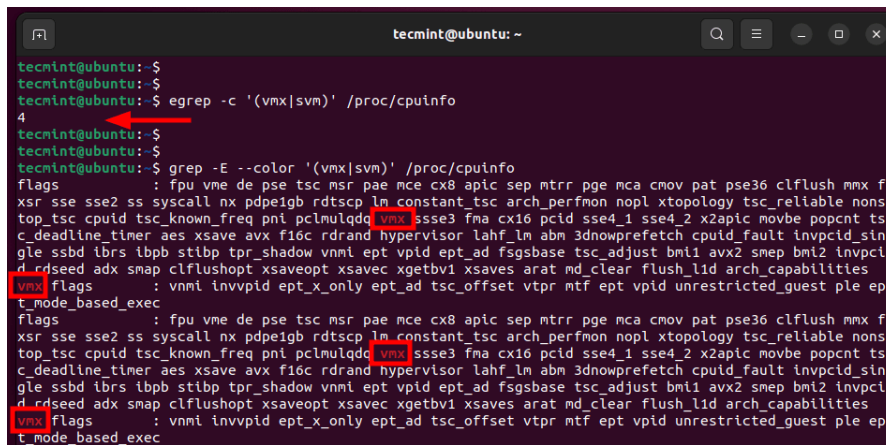
2. แบบที่ลงบน Linux (Ubuntu)

2.1 ขั้นตอนการลง QEMU/KVM on Ubuntu (ในกรณีที่ต้องใช้ ARM64 Arch)

Step 1: Check Virtualization Enabled in Ubuntu

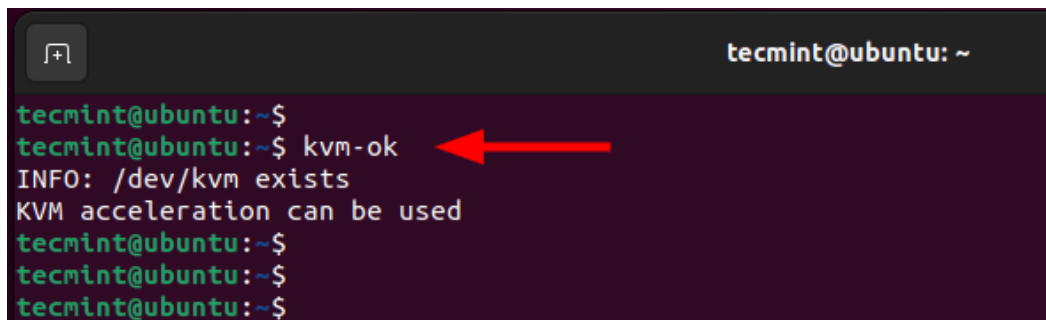
```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

```
$ grep -E --color '(vmx|svm)' /proc/cpuinfo
```



```
tecmint@ubuntu:~$  
tecmint@ubuntu:~$  
tecmint@ubuntu:~$ egrep -c '(vmx|svm)' /proc/cpuinfo  
4  
tecmint@ubuntu:~$  
tecmint@ubuntu:~$  
tecmint@ubuntu:~$ grep -E --color '(vmx|svm)' /proc/cpuinfo  
flags              : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx f  
xsr sse sse2 ss syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon nopl xtopology tsc_reliable nons  
top_tsc cpuid tsc_known_freq pni pclmulqdq vmx ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt ts  
c_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch cpuid_fault invpcid_sin  
gle ssbd ibrs ibpb stibp tpr_shadow vnmi ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 invpci  
d_rdtseed adx smap clflushopt xsaveopt xsavec xgetbv1 xsaves arat md_clear flush_l1d arch_capabilities  
vmx flags          : vmni invvpid ept_x_only ept_ad tsc_offset vtptr mtf ept vpid unrestricted_guest ple ep  
t_mode_based_exec  
flags              : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx f  
xsr sse sse2 ss syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon nopl xtopology tsc_reliable nons  
top_tsc cpuid tsc_known_freq pni pclmulqdq vmx ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt ts  
c_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch cpuid_fault invpcid_sin  
gle ssbd ibrs ibpb stibp tpr_shadow vnmi ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 invpci  
d_rdtseed adx smap clflushopt xsaveopt xsavec xgetbv1 xsaves arat md_clear flush_l1d arch_capabilities  
vmx flags          : vmni invvpid ept_x_only ept_ad tsc_offset vtptr mtf ept vpid unrestricted_guest ple ep  
t_mode_based_exec
```

```
$ kvm-ok
```



```
tecmint@ubuntu:~$  
tecmint@ubuntu:~$ kvm-ok  
INFO: /dev/kvm exists  
KVM acceleration can be used  
tecmint@ubuntu:~$  
tecmint@ubuntu:~$  
tecmint@ubuntu:~$
```

Step 2: Install QEMU/KVM on Ubuntu

```
$ sudo apt update
```

```
$ sudo apt install qemu-kvm virt-manager virtinst libvirt-clients bridge-utils libvirt-  
daemon-system -y
```

```
$ sudo systemctl enable --now libvirtd
```

```
$ sudo systemctl start libvirtd
```

\$ sudo systemctl status libvirtd

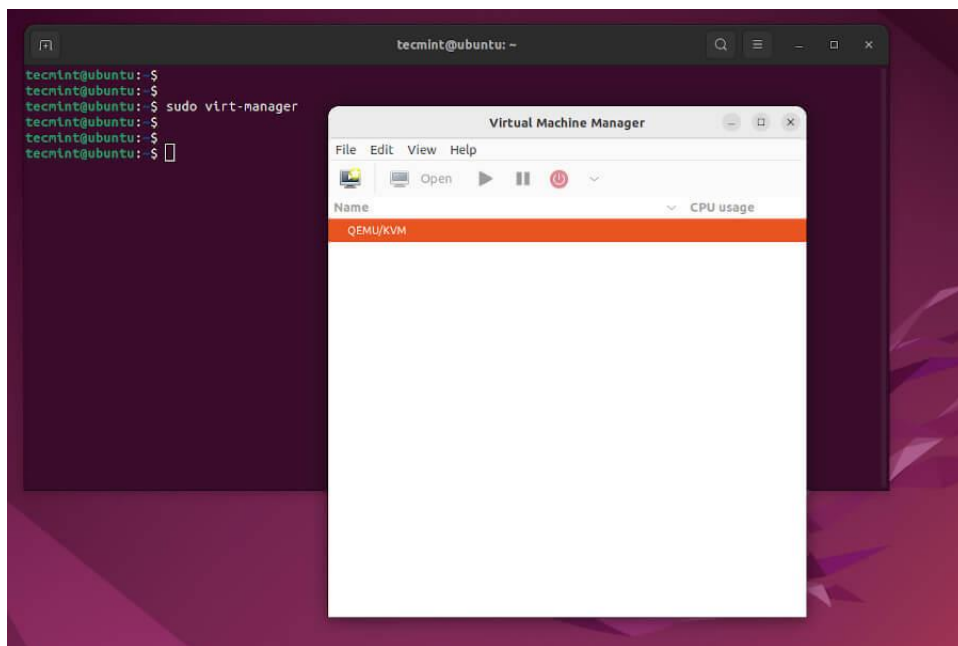
```
tecmin@ubuntu: ~  
tecmin@ubuntu:~$  
tecmin@ubuntu:~$ sudo systemctl enable --now libvirtd  
tecmin@ubuntu:~$  
tecmin@ubuntu:~$ sudo systemctl start libvirtd  
tecmin@ubuntu:~$  
tecmin@ubuntu:~$ sudo systemctl status libvirtd  
● libvirtd.service - Virtualization daemon  
   Loaded: loaded (/lib/systemd/system/libvirtd.service; enabled; vendor preset: enabled)  
   Active: active (running) since Tue 2022-10-25 11:49:16 EDT; 5min ago  
TriggeredBy: ● libvirtd-ro.socket  
              ● libvirtd.socket  
              ● libvirtd-admin.socket  
   Docs: man:libvirtd(8)  
          https://libvirt.org  
  Main PID: 11582 (libvirtd)  
    Tasks: 21 (limit: 32768)  
   Memory: 12.8M  
      CPU: 6.416s
```

\$ sudo usermod -aG kvm \$USER

\$ sudo usermod -aG libvirt \$USER

Step 3: Launch Virtual Machine Manager in Ubuntu

\$ sudo virt-manager



ตั้งแต่ Step4 เป็นต้นไปทำตามลิงก์ ด้านล่างได้เลย

ทำตามลิงก์ <https://www.tecmint.com/install-qemu-kvm-ubuntu-create-virtual-machines/>

การตั้งค่าเริ่มต้นของ Virtual Machine Manager

RAM = 2048 CPU = 2

2.2 ขั้นตอนการติดตั้ง Conda (กรณีที่ใช้ใน QEMU/ARM64)

ต้องไปที่ Virtual Machine Manager และทำการ start โปเจคก่อน

1. sudo apt-get update

เพื่ออัปเดต ubuntu

2. curl -L -O "https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-\$(uname)-\$(uname -m).sh"

bash Miniforge3-\$(uname)-\$(uname -m).sh

การติดตั้ง miniconda

***** ถ้ายังใช้ conda ไม่ได้ให้ลอง sudo update ก่อน *****

2.(1) conda config --set allow_conda_downgrades true

2.(2) conda install conda=4.6.11

3. conda create -n kivy python==3.9

หรือ CONDA_VERBOSE=3 conda create -n kivy --repodata-fn=repodata.json --offline

เพื่อสร้าง env ขึ้นใหม่

4. Conda activate kivy

เพื่อเปิดใช้งาน env ที่เราเลือก

5. sudo apt-get install python3-pip

ติดตั้ง python3 pip

6. python3 -m pip install --upgrade pip

อัปเดต pip python ก่อนติดตั้ง kivy

7. sudo apt install python3-dev

ติดตั้ง python3

8. conda install kivy -c conda-forge

ติดตั้ง kivy

***** ถ้าติดตั้งได้ให้ไปขั้นตอนที่ 10 *****

***** ถ้าทำขั้นตอนที่ 8 ไม่ได้ให้ไปขั้นตอนที่ 9 *****

9. การติดตั้ง kivy

9.1 sudo add-apt-repository ppa:kivy-team/kivy

9.2 sudo apt-get update

9.3 sudo apt-get install python3-kivy

10. df -h --total

ตรวจสอบพื้นที่ของ ubuntu

การใช้งาน Conda

- conda config --add channels conda-forge
- conda config --set channel_priority strict
- conda install gst-plugins-base gst-plugins-good gstreamer
- conda search gst-plugins-base --channel conda-forge

การติดตั้ง Kivy

- sudo add-apt-repository ppa:kivy-team/kivy
- sudo apt-get update
- sudo apt-get install python3-kivy
- sudo apt-get install kivy-examples
- sudo apt-get install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev libgstreamer-plugins-bad1.0-dev gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-bad gstreamer1.0-plugins-ugly gstreamer1.0-libav gstreamer1.0-tools gstreamer1.0-x gstreamer1.0-alsa gstreamer1.0-gl gstreamer1.0-gtk3 gstreamer1.0-qt5 gstreamer1.0-pulseaudio

Install Libraries

1. ติดตั้ง Play Video

- `sudo apt-get install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev libgstreamer-plugins-bad1.0-dev gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-bad gstreamer1.0-plugins-ugly gstreamer1.0-libav gstreamer1.0-tools gstreamer1.0-x gstreamer1.0-alsa gstreamer1.0-gl gstreamer1.0-gtk3 gstreamer1.0-qt5 gstreamer1.0-pulseaudio`

2. ติดตั้ง opencv-python

- `pip install opencv-python`

3. ติดตั้ง pyjnius

- `pip install pyjnius`

4. ติดตั้ง numpy

- `pip install numpy`

5. ติดตั้ง pyproject-toml

- `pip install pyproject-toml`

6. ติดตั้ง kivy

6.1 `pip install kivy`

6.2 `python -m pip install kivy` และ `python -m pip install fpyplayer` ใช้ในกรณีที่ 7.1 ติดตั้งไม่สำเร็จ

6.3 `conda install kivy -c conda-forge` ใช้ในกรณีที่ 7.1 และ 7.2 ติดตั้งไม่สำเร็จ

7. ติดตั้ง buildozer

`pip install buildozer`

8. ติดตั้ง cython

`pip install cython`

9. ติดตั้ง python-for-android

`pip install python-for-android`

10. ติดตั้ง gstreamer-player

pip install gstreamer-player

11. ติดตั้ง **cmake**

pip install cmake

12. ติดตั้ง **mediapipe**



pip install mediapipe

13. ติดตั้ง **unzip**

sudo apt install unzip

14. ติดตั้ง **java เพื่อ build**

sudo apt-get install openjdk-8-jdk

15. sudo apt-get install build-essential libssl-dev

วิธีการ Build & Compile App ด้วย Buildozer/Kivy

1. ใช้คำสั่ง Buildozer init เพื่อเริ่มการ Build
2. จะได้ Text File ชื่อ Buildozer.spec โดยเราจะต้องเข้าไปแก้ config ใน buildozer.spec ดังนี้

```
# (list) Source files to include (let empty to include all the files)
```

```
source.include_exts = py,png,jpg,kv,atlas,mp4
```

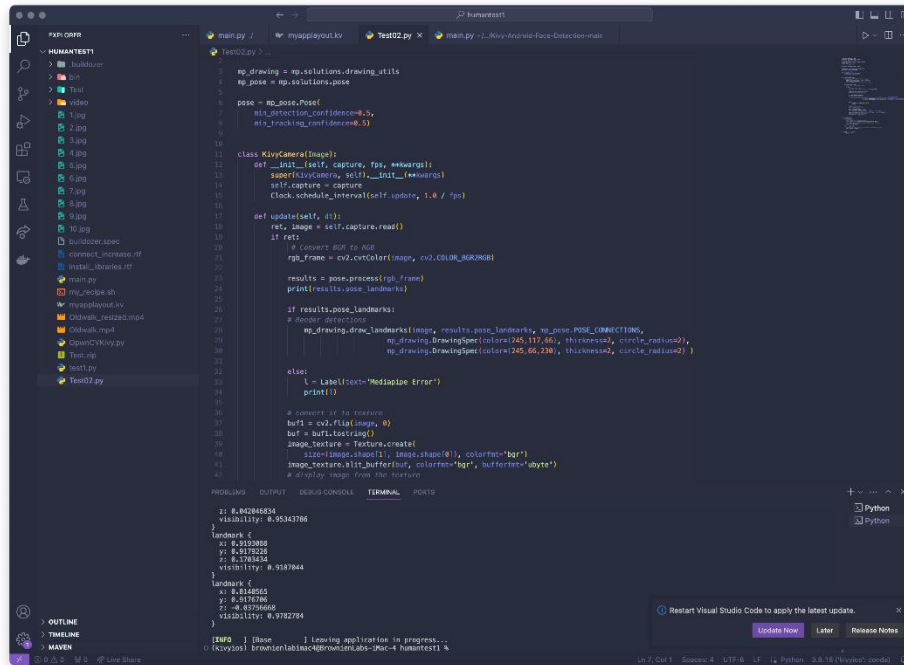
```
# (list) Application requirements
```

```
# comma separated e.g. requirements = sqlite3,kivy
```

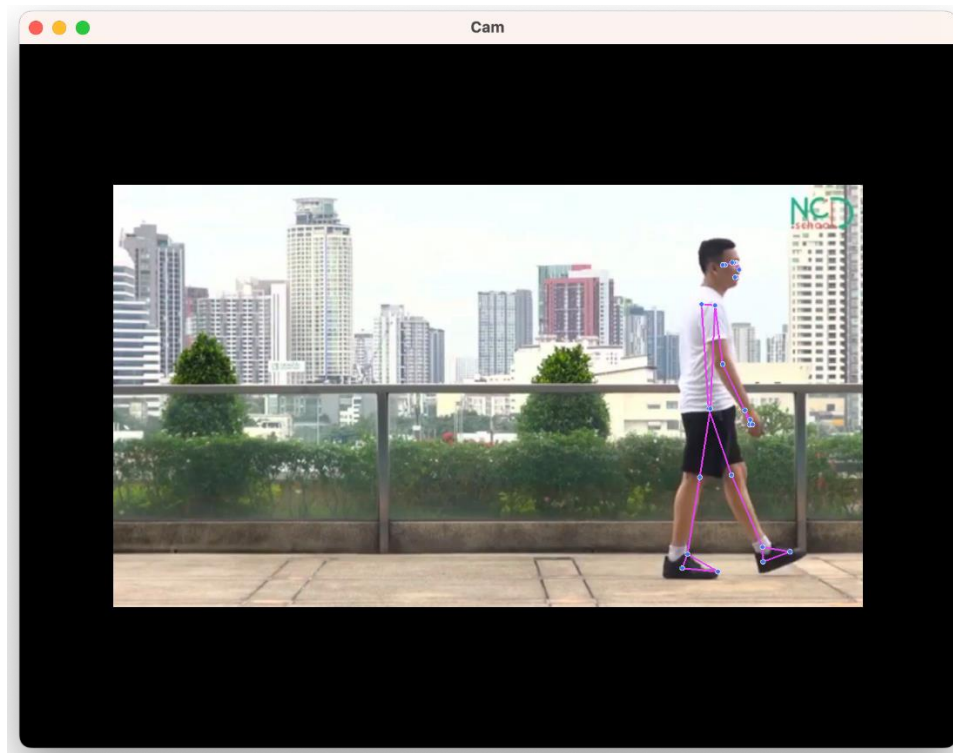
```
requirements = python3,kivy,opencv,numpy,mediapipe
```

3. buildozer android debug deploy run เพื่อทำการ Build Application

ตัวอย่างงานที่ทำ

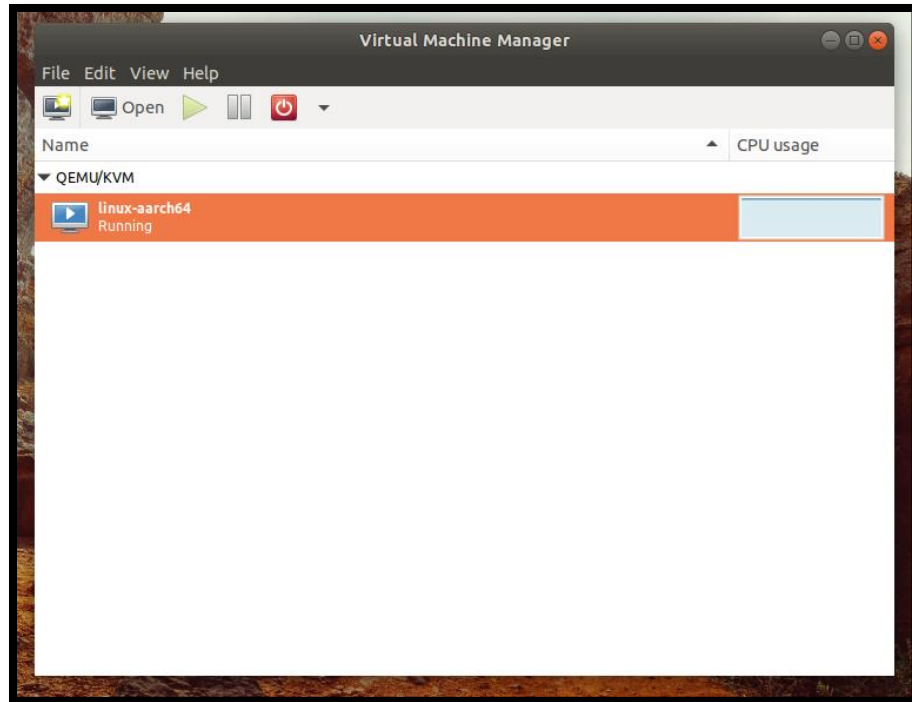


ภาพที่ 1 ตัวอย่าง Code

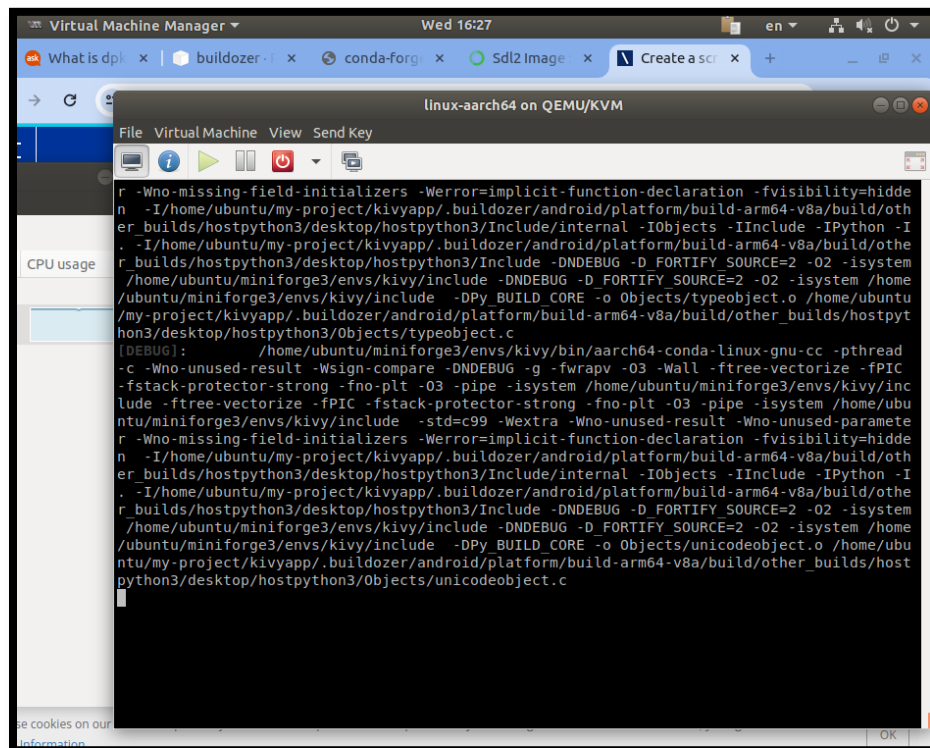


ภาพที่ 2 ตัวอย่างการรัน Code

?



ภาพที่ 3 QEMU



ภาพที่ 4 ตัวอย่างการทำงานของ QEMU

ปัญหาที่พบ

bug คือ OSError: [Error8] Exec format error: '/home/ubuntu/.buildozer/android/platform/android-sdk/build-tools/34.0.0/aidl'

ขั้นตอนการแก้ไข bug

1. sudo apt-get install aidl
2. which aidl /usr/bin/aidl
3. cp /usr/bin/aidl ~/.buildozer/android/platform/android-sdk/build-tools/34.0.0/
4. chmod 777 ~/.buildozer/android/platform/android-sdk/build-tools/34.0.0/aidl

จากลิงก์ ดังนี้ <https://stackoverflow.com/questions/44836469/aidl-not-found-please-install-it/68654306#68654306>

bug คือ configure: error: c compiler cannot create executables see config.log' for more details

แก้ไข bug ด้วยคำสั่ง conda install -c conda-forge libffi

จากลิงก์ ดังนี้ <https://anaconda.org/conda-forge/libffi>

bug คือ early EOF index-pack failed

Prebuilding sdl_image for arm64-v8a

แก้ไข bug ด้วยคำสั่ง sudo apt-cache search libsdl2-image

หรือ sudo apt-get install libsdl2-image-dev

จากลิงก์ ดังนี้

https://lazyfoo.net/tutorials/SDL/06_extension_libraries_and_loading_other_image_formats/linux/index.php