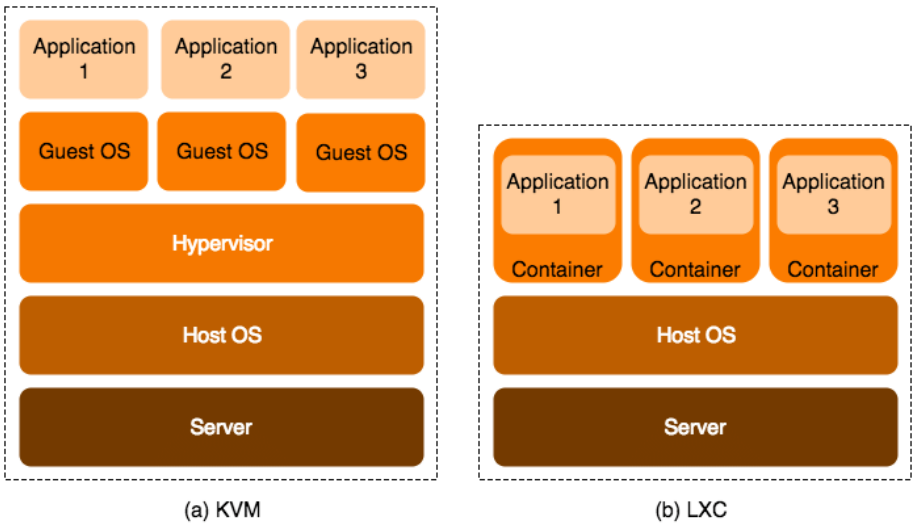
I. 基本概念

目前的虛擬技術可以概分為「全虛擬化」、「半虛擬化」與「作業系統層虛擬化」的三大主流技術，個別的概念如下：

1. 全虛擬化（Full Virtualization）：主機端（Host)藉由軟體或硬體輔助支援模擬完整的底層硬體環境，讓客戶機(Guest)作業系統以為自己運作在一般電腦，如Linux KVM。
2. 半虛擬化（Para-Virtualization）：主機端（Host)不需要模擬底層硬體環境，而是將一部分硬體介面直接提供給客戶機(Guest)作業系統，如：Xen。
3. 作業系統層虛擬化（OS-level virtualization）：主機端（Host)直接將作業系統核心虛擬化，因此客戶機(Guest)無須也無法安裝自己的作業系統，客戶機(Guest)又稱為容器(Container)，每個容器的執行程序相互獨立，因此對於使用者來說，就像是在使用自己的虛擬機，如：LXC與Docker。

目前伺服器上支援的種類分別為「全虛擬化」的Linux KVM與「作業系統層虛擬化」的LXC，兩者的比較如以下的示意圖。圖中可以看到兩者最大的差異在於，KVM有多兩層：Hypervisor與Guest OS，其中Hypervisor是用來模擬客戶機運行的底層硬體環境，因此客戶機還需要再安裝自己的作業系統(Guest OS)。此種方式的優勢在於客戶端與本機端屬於完全隔離，因此客戶端可以安裝任何支援x86 CPU的作業系統，例如：Windows、Linux、FreeBSD等，不過隨之而來的缺點就是效能的減損。

LXC透過容器(Container)技術，免除Hypervisor與Guest OS的效能減損，優點在於主機端與客戶端共享同一個作業系統核心，因此兩者之間可以達成更好的資源共享，但缺點就是無法選擇客戶端的作業系統(Guest OS)，僅能使用Linux。



https://www.researchgate.net/figure/KVM-and-LXC-see-online-version-for-colour_fig1_321297465

再以GPU來說，為了提昇客戶機運算存取的效率，目前的虛擬技術多半都提供「穿透技術」(Passthrough)，讓客戶機可以直接存取GPU的硬體，無須經由軟體模擬，盡可能減低虛擬技術造成的效能減損。

KVM與LXC在技術上的差異，造成兩者在GPU的分享上也有很大的差別。KVM的運作模式下，目前僅有Nvidia最高等級的GPU支援KVM的虛擬化(如：Tesla系列)，又稱為vGPU，可以將一片GPU分享給不同的KVM客戶端使用。其餘等級的GPU每一片僅能分配給一台客戶機使用，其他客戶機完全看不到也無法使用未被分配的GPU，若其他客戶機需要使用GPU，只能把某片GPU從使用中的客戶機切換過來。而LXC的運作模式下，由於主機端與客戶端兩者共享作業核心，因此無論哪個等級的GPU都可以讓全部的客戶端共享，不過Nvidia應然有限制每片GPU最多可以跑幾個程序。

II. 安裝與流程

1. 請勿在客戶機自行安裝Nvidia Driver(驅動程式)

LXC運作模式下，由於本機端與客戶機共享作業系統核心，因此本機端與客戶端的Nvidia Driver版本必須完全一致，因此當你收到客戶機時，都已安裝好與本機端一致的驅動程式，可以透過以下的指令(黃色部份)來驗證驅動程式是否已經裝好。

```
# nvidia-smi
```

若正常運作，會看到類似以下的畫面：

```
(base) connie@connie-lxc:~$ nvidia-smi
Thu Aug 12 16:12:16 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.57.02    Driver Version: 470.57.02    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  Off  | 00000000:1D:00.0 Off |                  N/A |
|  0%   39C    P8    22W / 350W |      0MiB / 24268MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA GeForce ...  Off  | 00000000:20:00.0 Off |                  N/A |
|  0%   38C    P8    18W / 350W |      0MiB / 24268MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  NVIDIA GeForce ...  Off  | 00000000:21:00.0 Off |                  N/A |
| 44%   53C    P2    71W / 250W |  10898MiB / 11019MiB |     25%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   3  Tesla V100S-PCI...  Off  | 00000000:24:00.0 Off |                    0 |
| N/A   42C    P0    26W / 250W |      0MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

若沒有看到以上的畫面，請移除現在安裝的驅動程式，並用以下指令(黃色部份)重新安裝。

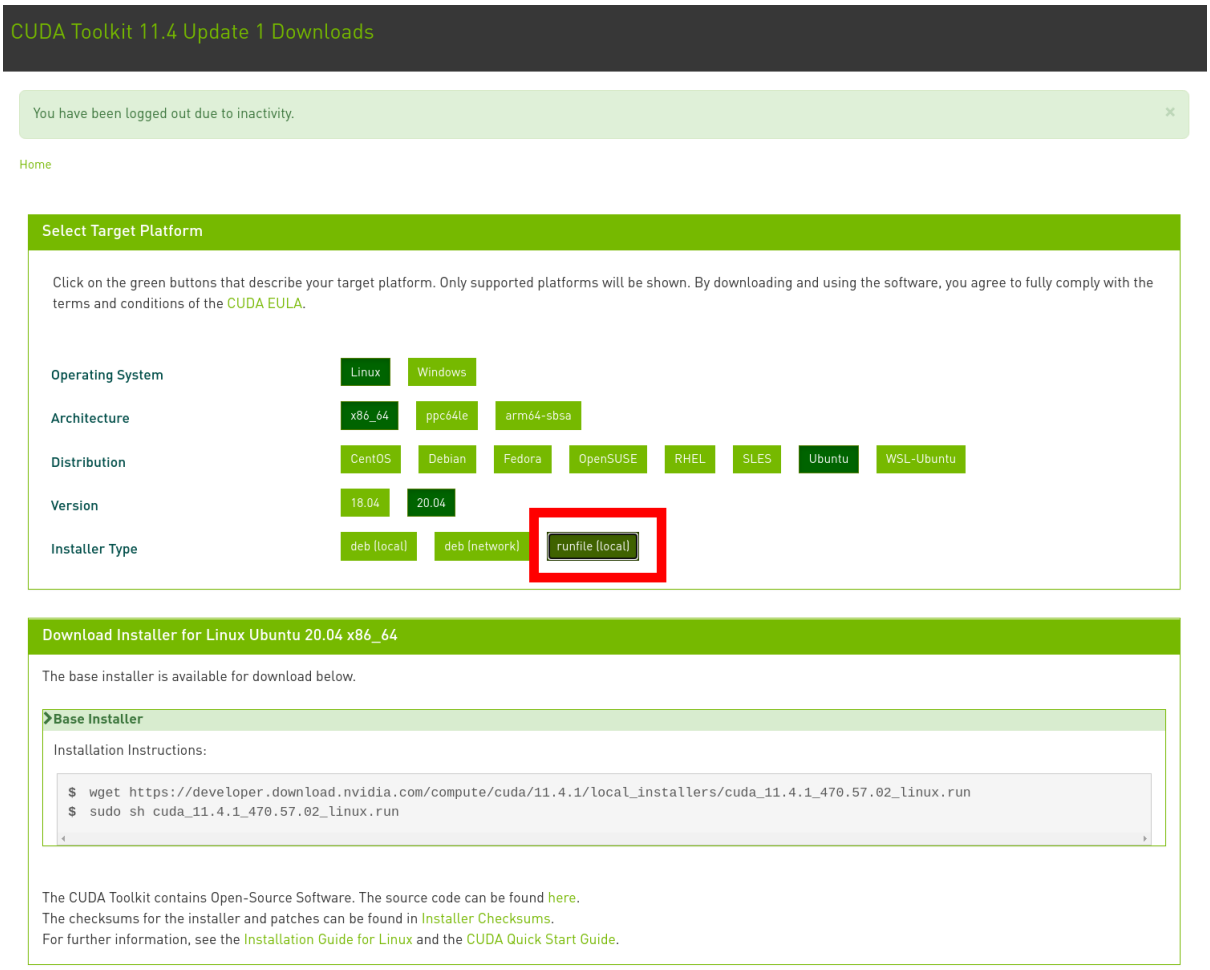
```
# sudo /root/NVIDIA-Linux-x86_64-470.57.02.run --no-kernel-module
```

2. 下載安裝CUDA

某些機器學習套件會指定支援的CUDA版本，請先確認要安裝的版本後，搜尋並進入nvidia網站下載相對應的套件。以下以版本11.4作為範例，進入對應版本

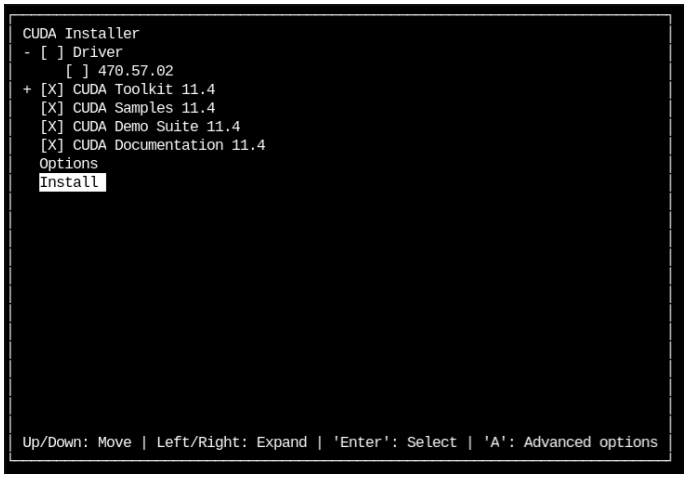
的下載網頁之後，請下載「runfile(local)」，下載與安裝指令(黃色部份)如下。

注意：請勿下載deb檔案，也不要用apt安裝ubuntu提供的套件，因為兩者都安裝附帶的驅動程式，造成原先的驅動程式被覆蓋。




```
# wget https://developer.download.nvidia.com/compute/cuda/11.4.1/local_installers/cuda_11.4.1_470.57.02_linux.run
# sudo sh cuda_11.4.1_470.57.02_linux.run
```

安裝過程會出現以下的選單，請勿勾選驅動程式。



安裝後設置環境參數(黃色部份)如下：

```
# echo '# CUDA Soft Link' >> ~/.bashrc
# echo 'export PATH=/usr/local/cuda-11.4/bin${PATH:+:${PATH}}' >> ~/.bashrc
# echo 'export LD_LIBRARY_PATH=/usr/local/cuda-11.4/lib64${LD_LIBRARY_PATH:+:
# ${LD_LIBRARY_PATH}}' >> ~/.bashrc
# source ~/.bashrc
```

最後透過以下指令(黃色部份)測試CUDA是否可以順利抓到GPU，若正常應該會看到以下畫面。

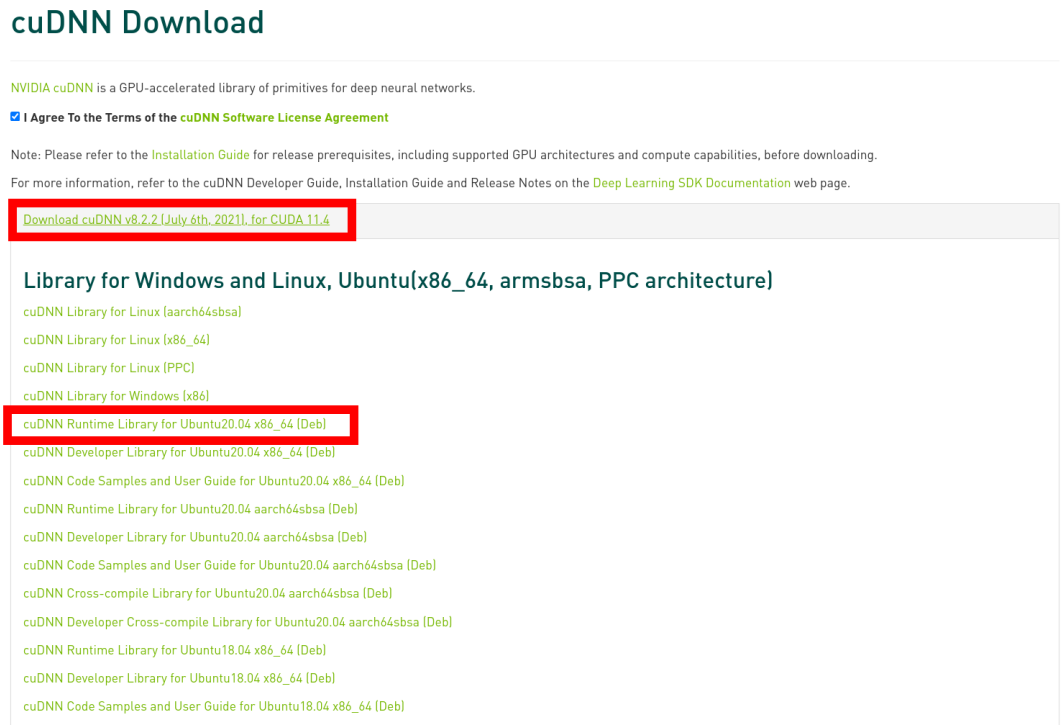
```
# cd /usr/local/cuda/samples/1_Utilities/deviceQuery
# make
# ./deviceQuery
```

```
Device 3: "NVIDIA GeForce RTX 2080 Ti"
  CUDA Driver Version / Runtime Version          11.4 / 11.4
  CUDA Capability Major/Minor version number:    7.5
  Total amount of global memory:                 11019 MBytes (11554717696 bytes)
  (068) Multiprocessors, (064) CUDA Cores/MP:    4352 CUDA Cores
  GPU Max Clock rate:                            1545 MHz (1.54 GHz)
  Memory Clock rate:                             7000 Mhz
  Memory Bus Width:                              352-bit
  L2 Cache Size:                                 5767168 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total shared memory per multiprocessor:        65536 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  1024
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 3 copy engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Managed Memory:                Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Supports MultiDevice Co-op Kernel Launch:      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 33 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >
> Peer access from NVIDIA GeForce RTX 3090 (GPU0) -> NVIDIA GeForce RTX 3090 (GPU1) : No
> Peer access from NVIDIA GeForce RTX 3090 (GPU0) -> Tesla V100S-PCIE-32GB (GPU2) : No
> Peer access from NVIDIA GeForce RTX 3090 (GPU0) -> NVIDIA GeForce RTX 2080 Ti (GPU3) : No
> Peer access from NVIDIA GeForce RTX 3090 (GPU1) -> NVIDIA GeForce RTX 3090 (GPU0) : No
> Peer access from NVIDIA GeForce RTX 3090 (GPU1) -> Tesla V100S-PCIE-32GB (GPU2) : No
> Peer access from NVIDIA GeForce RTX 3090 (GPU1) -> NVIDIA GeForce RTX 2080 Ti (GPU3) : No
> Peer access from Tesla V100S-PCIE-32GB (GPU2) -> NVIDIA GeForce RTX 3090 (GPU0) : No
> Peer access from Tesla V100S-PCIE-32GB (GPU2) -> NVIDIA GeForce RTX 3090 (GPU1) : No
> Peer access from Tesla V100S-PCIE-32GB (GPU2) -> NVIDIA GeForce RTX 2080 Ti (GPU3) : No
> Peer access from NVIDIA GeForce RTX 2080 Ti (GPU3) -> NVIDIA GeForce RTX 3090 (GPU0) : No
> Peer access from NVIDIA GeForce RTX 2080 Ti (GPU3) -> NVIDIA GeForce RTX 3090 (GPU1) : No
> Peer access from NVIDIA GeForce RTX 2080 Ti (GPU3) -> Tesla V100S-PCIE-32GB (GPU2) : No

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 11.4, CUDA Runtime Version = 11.4, NumDevs = 4
Result = PASS
```

3. 下載安裝cuDNN

   請先確認要安裝cuDNN的版本後，搜尋並進入nvidia網站下載相對應的套件。以下以版本8.2作為範例，進入對應版本的下載網頁之後，請點選搭配CUDA版本的連結，之後下載「cuDNN Runtime Library for Ubuntu 20.04 x86_64 (Deb)」，最後透過以下指令進行安裝。




```
# sudo dpkg -i /PATH/TO/libcudnn8_8.2.2.26-1+cuda11.4_amd64.deb
```

4. 安裝機器學習套件

   後續請安裝需要使用的機器學習套件，如：Tensorflow。