Pre ML Server Setup

Part 1: BIOS/UEFI Configuration:
(Enter BIOS/UEFI settings before installing Ubuntu)

Step 1: Enter BIOS Setup:

1. Power on or reboot the server.
2. When prompted (usually within a few seconds), press `F9` to enter **System Utilities** (BIOS setup).
3. You may also see a prompt for iLO – skip that for now unless you want to configure remote access.

Step 2 : Enable Virtualization (CPU):

In BIOS settings:

1. Navigate to:
   `System Configuration` > `BIOS/Platform Configuration (RBSU)` > `Processor Options`
2. Enable the following options:
   a. **AMD-V / Intel VT-x**: Enable (this allows virtualization extensions)
   b. **SVM Mode** (for AMD CPUs): Enable
   c. **SR-IOV**: Enable (optional, for networking virtualization)

Step 3 : Enable IOMMU / PCIe Passthrough (for GPU Virtualization) :

Still under `Processor Options`:

1. Enable:

   a.**IOMMU**: Enable (Required for PCIe passthrough)
   b.**Memory Mapped I/O above 4GB**: Enable (important for large GPUs)
   c.**ACS Support** (optional but helpful for passthrough isolation): Enable

Step 4 : GPU Options :

If the server has discrete GPUs:

1. Go to:

`System Configuration` > `BIOS/Platform Configuration (RBSU)` > `Advanced Options` > `PCIe Options`

2. Ensure:

   - ○ **PCIe ARI Support**: Enabled
   - ○ **Above 4G Decoding**: Enabled (critical for GPU memory mapping)

Step 5 : Boot Settings :

To allow boot from USB/DVD:

1. Go to:

`System Configuration` > `BIOS/Platform Configuration (RBSU)` > `Boot Options`

2. Ensure:

- **UEFI Boot Mode**: Enabled
- Set **USB or DVD** as first boot device.

Optional: Disable Secure Boot if needed under `Secure Boot` settings.

Part 2: Ubuntu 22.04 Installation:

Step 6: Create a Bootable USB with Ubuntu:

On another computer:

1. Download Ubuntu 22.04 ISO from ubuntu.com.

2. Use **Rufus** (Windows) or **balenaEtcher** (macOS/Linux) to create a bootable USB.

Step 7 : **Boot from USB**

1. Insert the bootable USB into the HPE server.
2. Power on and press F11 to select **One-Time Boot Menu**.
3. Select the USB drive.
4. Ubuntu installer will start.

Step 8: Ubuntu Server Installation:

1. Select language and keyboard layout.
2. Choose "Install Ubuntu Server."
3. Configure:

- Network (static or DHCP)
- Disk Partitioning: Use entire disk (unless custom layout needed)
- Profile (name, username, password)

4. Select optional snaps (you can skip).
5. Installation begins. Reboot after it's done.

Step 9: Post-Install Configuration:

After reboot:

1. Login with your user credentials.

2. Update packages:
    - sudo apt update && sudo apt upgrade -y

Part 3: Verify Virtualization Support:

Step 10: Check Virtualization:

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

  Output should be **1 or more** (means VT-x/SVM is available)

- To verify IOMMU:

  - dmesg | grep -i iommu

Technical Terms :

PCIe:

PCIe is a **high-speed serial computer expansion bus standard** that connects high-performance components—like GPUs, SSDs, and network cards—to the motherboard.

Think of it as:

A **data highway** between your CPU and critical hardware like the GPU or NVMe SSD.

Key Concepts:

1. Lanes (x1, x4, x8, x16)

- Each **lane** consists of two pairs of wires: one for sending data, one for receiving.

- **More lanes = more bandwidth**.

- For example:
    - x1 → 1 lane (used for WiFi cards)
    - x4 → 4 lanes (used for NVMe SSDs)
    - x16 → 16 lanes (used for GPUs)

2. **Versions (v1.0 → v5.0 and beyond)**

Each newer version increases data rate:

- PCIe 3.0 (commonly used): ~1 GB/s per lane
- PCIe 4.0: ~2 GB/s per lane
- PCIe 5.0: ~4 GB/s per lane

A **PCIe 4.0 x16 GPU** can transfer up to **32 GB/s**!

What connects via PCIe?

- GPUs (NVIDIA, AMD)
- NVMe SSDs
- Network Interface Cards (10GbE, 25GbE, etc.)
- RAID controllers
- Capture cards

Why is PCIe Important for Virtualization?

- For GPU Passthrough or SR-IOV:
    1. PCIe allows direct communication between a **VM and a physical device** (e.g., GPU).
    2. Technologies like **IOMMU** rely on PCIe to isolate and safely share devices.

Example:

Your GPU is plugged into a **PCIe x16 slot**. If you're doing GPU passthrough to a VM, you're allowing that VM to directly use the GPU via the PCIe interface.

BIOS Settings :

1. **AMD-V / Intel VT-x** (a.k.a. Virtualization Extensions)
   - **Purpose**: Allows the CPU to support running virtual machines with near-native performance.

   - **Technical term**: These are hardware-assisted virtualization features.
   - **Intel** calls it **VT-x**, and **AMD** calls it **AMD-V**.
   - **Why enable it**: Required by hypervisors like KVM, VirtualBox, VMware, etc.
2. **SVM Mode** (Secure Virtual Machine)
   - **Specific to AMD processors**.
   - **Purpose**: Enables AMD-V instructions at the BIOS level.
   - **Why enable it**: Without this, AMD CPUs can't run virtual machines.
3. SR-IOV (Single Root I/O Virtualization)
   - **Purpose**: Allows a physical PCIe device (like a NIC or GPU) to appear as multiple **virtual** devices to VMs.
   - **Technical use**: Essential in high-performance environments, especially in cloud and network virtualization.
   - **Why enable it**: Lets VMs directly access slices of physical devices for performance.

4. **IOMMU (Input-Output Memory Management Unit)**
   - **Intel** calls it **VT-d**, **AMD** calls it **IOMMU**.
   - **Purpose**: Allows direct access of PCIe devices (e.g., GPUs) to virtual machines.
   - **Why enable it**: Needed for **PCI passthrough** – giving a full GPU or NIC to a VM.

5. Memory-Mapped I/O above 4GB

- **Purpose**: Allows the system to assign memory address space to PCIe devices above 4 GB, which is necessary for 64-bit addressable devices like modern GPUs.
- **Why enable it**: Prevents resource conflicts and allows full GPU memory access.

6. ACS (Access Control Services)
    - **Purpose**: Helps isolate PCIe devices into separate IOMMU groups.
    - **Why it matters**: When doing GPU passthrough, ACS makes sure each device can be safely assigned to a different VM.
7. PCIe ARI (Alternative Routing-ID Interpretation)
    - **Purpose**: Enhances how many PCIe functions a single physical slot can support.
    - **Why enable it**: Helps with SR-IOV and multi-function devices like GPUs with multiple display pipelines or NICs with virtual functions.
8. Above 4G Decoding
    - **Purpose**: Allows the use of large address spaces (>4GB) for PCIe devices like GPUs.
    - **Why enable it**: Critical for systems with multiple GPUs or high-memory GPUs (e.g., 16GB, 24GB).
9. UEFI Boot Mode
    - **UEFI** is the modern replacement for legacy BIOS.
    - **Why enable it**: Required by modern OSes like Ubuntu 22.04 for features like Secure Boot and faster booting.
10. **Secure Boot**
    - **Purpose**: Prevents unauthorized OS or bootloaders from running.
    - **Why you might disable it**: If you’re using unsigned drivers (e.g., NVIDIA proprietary driver), Secure Boot might block them.

General Server Statics Commands:

1. All PCI Devices (GPUs, NICs, etc.)
    - lspci

      Lists all devices connected to **PCIe/PCI** slots.

      Add `-v` or `-vv` for more details:

        - lspci -nnv

2. All USB Devices

```
lsusb
```

- Shows devices connected via USB ports.

3. Storage Devices (Disks, SSDs)

```
lsblk
```

- Lists block devices (like `/dev/sda`, `/dev/nvme0n1`).

```
sudo fdisk -l
```

- Detailed partition info.

4. Network Interfaces

```
ip link show
```

- Shows all network interfaces.

```
lshw -class network
```

- Detailed info about each network device.

---

5. Detailed Hardware Info (All Devices)

```
sudo lshw
```

- Full system hardware summary.
- To view a specific class, use:

```
sudo lshw -class display    # For GPU
sudo lshw -class processor  # For CPU
```

---

6. CPU & Memory Info

```
lscpu          # Detailed CPU architecture
```

```
free -h          # Memory usage
```

---

7. IOMMU / PCI Grouping (for passthrough)

```
find /sys/kernel/iommu_groups/ -type l
```

- Lists IOMMU groups, helpful for GPU passthrough.

Bonus: View All Detected Devices

```
dmesg | less
```

- Kernel logs that include detected hardware during boot.