Environment setup:

1. **WSL & ubuntu**

   WSL = Windows Subsystem for Linux
   It is a feature in Windows that allows you to run a Linux environment directly inside Windows, without the overhead of a full virtual machine. WSL lets you run Linux command-line tools, utilities, and applications natively on Windows.
   What is Ubuntu?
   Ubuntu is a popular Linux distribution (distro). Think of it as a flavor/version of Linux, like Windows has different editions. When you do wsl --install, it installs Ubuntu as your Linux environment by default.

   Enable WSL and Virtual Machine Platform
   Open PowerShell as Administrator and run:

   ```
   wsl --install
   ```

   Set WSL version to 2:

   ```
   wsl --set-default-version 2
   ```

2. **Move to disk D**

   ```
   mkdir D:\WSL\Ubuntu
   wsl --import Ubuntu D:\WSL\Ubuntu <path-to-ubuntu-rootfs.tar> --version 2
   ```

   Check:

   ```
   Get-ChildItem -Path "HKCU:\Software\Microsoft\Windows\CurrentVersion\Lxss" |
     ForEach-Object {
       Get-ItemProperty -Path $_.PSPath |
       Select-Object DistributionName, BasePath
     }
   ```

   Gives:

   ```
   DistributionName : Ubuntu
   BasePath         : D:\WSL\Ubuntu
   ```

   Check:

   ```
   wsl --list --verbose
   ```

   Gives:

   ```
     NAME      STATE           VERSION
   * Ubuntu    Running         2
   ```

3. **Work in ubuntu**

   ```
   Search ubuntu -> open terminal
   OR in powershell -> wsl -d Ubuntu
   ```

   Commands:

   ```
   whoami          # shows your Linux username
   ```

```
ls              # list files
pwd               # print working directory
cd ~             # go to home directory
```

Update package manager:

```
sudo apt update && sudo apt upgrade -y
```

Install Python and pip:

```
sudo apt install -y python3 python3-pip
```

Check:

```
python3 --version
pip3 --version
```

4. Set up a project environment (virtual environment in ubuntu)

Tool	Command Style	Notes
`venv + pip`	`python3 -m venv venv && pip install ...`	Simple and works, but you must pick/install the right PyTorch manually
`uv venv`	`uv venv --python 3.12 --seed` + `uv pip install ...`	Handles CUDA automatically, faster pip replacement
`conda`	`conda create -n ...` + `conda install ...`	Heavier, but well-known and GUI-friendly

- Use python’s ven in ubuntu

  Install python3 & venv

  ```
  sudo apt update
  sudo apt install -y python3 python3-pip python3-venv
  ```

  Create venv

  ```
  mkdir my_project
  cd my_project
  python3 -m venv venv
  ```

  Activate

  ```
  source venv/bin/activate
  ```

  Gives prompt: (venv) junrong@Junrong:~/my_project$

  Install packages in the environment

  ```
  pip install vllm torch numpy
  ```

- Use uv (recommended in vLLM)
  uv is a new, ultra-fast Python package manager and environment manager, created by the same team behind pdm and other modern Python tooling. uv combines the speed of:
  pip for package installation, venv for environment management, pip-tools for dependency resolution

  Summary:

  ```
  curl -Ls https://astral.sh/uv/install.sh | sh (install uv, just need once)
  mkdir new-project
  cd new-project
  uv venv --python 3.12 --seed
  source .venv/bin/activate
  uv pip install vllm --torch-backend=auto
  ```

  For my project:

  ```
  # Step 1: Install uv (just once)
  junrong@Junrong:~$ curl -Ls https://astral.sh/uv/install.sh | sh
  #restart shell
  # make new project, set the virtual env for the project
  junrong@Junrong:~$ mkdir llmInference-vllm
  junrong@Junrong:~$ cd llmInference-vllm/
  The folder is in: \\wsl.localhost\Ubuntu\home\junrong
  # create virtual env
  junrong@Junrong:~/llmInference-vllm$ uv venv --python 3.12 --seed
  # activate env
  junrong@Junrong:~/llmInference-vllm$ source .venv/bin/activate
  # install vLLM & packages
  (llmInference-vllm) junrong@Junrong:~/llmInference-vllm$ uv pip install vllm --torch-backend=auto
  ```

5. **Install NVIDIA GPU driver for WSL**
   In ubuntu bash

   ```
   nvidia-smi
   ```

6. **Install CUDA toolkits**

   ```
   sudo apt update
   sudo apt install -y cuda
   ```

7. **Installation check**
   Check installation: `python -c "import vllm; print(vllm.__version__)"`
   Run simple inference:

   ```
   python3 -m vllm.entrypoints.openai.api_server --model meta-llama/Llama-2-7b-hf
   ```

```
curl http://localhost:8000/v1/completions \
  -H "Content-Type: application/json" \
  -d '{
    "model": "meta-llama/Llama-2-7b-hf",
    "prompt": "Once upon a time,",
    "max_tokens": 20
  }'
```

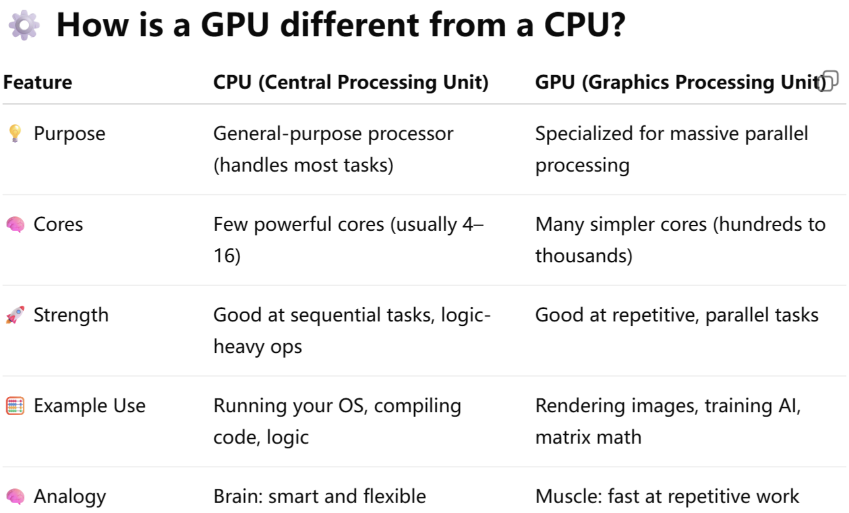
Inference Code:

** everytime when opening new ubuntu terminal, need to activate the virtual env

```
junrong@Junrong:~$ cd llmInference-vllm/
junrong@Junrong:~/llmInference-vllm$ source .venv/bin/activate
-> (llmInference-vllm) junrong@Junrong
```

GPU:
GPU stands for Graphics Processing Unit.

Originally, GPUs were created to handle graphics rendering (e.g., in games). But over time, people realized GPUs are very good at running certain types of computations quickly — especially parallel operations used in machine learning.



⚙️ How is a GPU different from a CPU?

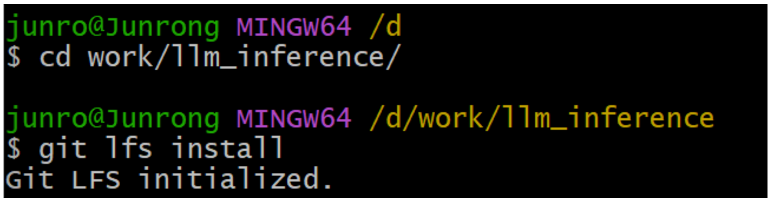
Feature	CPU (Central Processing Unit)	GPU (Graphics Processing Unit)
💡 Purpose	General-purpose processor (handles most tasks)	Specialized for massive parallel processing
🧠 Cores	Few powerful cores (usually 4–16)	Many simpler cores (hundreds to thousands)
🚀 Strength	Good at sequential tasks, logic-heavy ops	Good at repetitive, parallel tasks
🧮 Example Use	Running your OS, compiling code, logic	Rendering images, training AI, matrix math
🧠 Analogy	Brain: smart and flexible	Muscle: fast at repetitive work

LLMs like GPT, Qwen, etc., require: Matrix multiplication, Vector operations, Attention over many tokens -> These are very parallel operations — and GPUs are optimized for exactly that.

Basic script structure:
Create py file:
In the directory, nano run_inference.py (can also use vim)

Write the code
Online vs offline
Nano commands: Save and exit: Press Ctrl+O (save), then Enter, then Ctrl+X (exit).
Run: python3 run_inference.py

Running offline:
Download models:



```
junro@Junrong MINGW64 /d
$ cd work/llm_inference/

junro@Junrong MINGW64 /d/work/llm_inference
$ git lfs install
Git LFS initialized.
```

```
git lfs install
git clone https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct
git clone https://huggingface.co/Qwen/Qwen2.5-0.5B
git clone https://huggingface.co/Qwen/Qwen3-0.6B-Base
git clone https://huggingface.co/Qwen/Qwen3-0.6B
```

In ubuntu:

Copy paste to ubuntu

Update script to use local path

```
model_name = "/home/junrong/llmInference-vllm/Qwen2.5-0.5B-Instruct"
llm = LLM(model=model_name)
```