GPU support in Docker Desktop for Windows

- Prerequisites
- Validate GPU support
- Run a real-world model: SmolLM2 with Docker Model Runner

> This page covers GPU passthrough to Linux containers using `--gpus`, which is only available on Windows with the WSL 2 backend. Other forms of GPU acceleration in Docker Desktop, such as GPU-accelerated inference in Docker Model Runner, are available on other platforms.

Docker Desktop for Windows supports NVIDIA GPU Paravirtualization (GPU-PV) on NVIDIA GPUs, allowing containers to access GPU resources for compute-intensive workloads like AI, machine learning, or video processing.

Prerequisites

To enable WSL 2 GPU Paravirtualization, you need:

- A Windows machine with an NVIDIA GPU
- Up to date Windows 10 or Windows 11 installation
- Up to date drivers from NVIDIA supporting WSL 2 GPU Paravirtualization
- The latest version of the WSL 2 Linux kernel. Use `wsl --update` on the command line
- To make sure the WSL 2 backend is turned on in Docker Desktop

Validate GPU support

To confirm GPU access is working inside Docker, run the following:

```
$ docker run --rm -it --gpus=all nvcr.io/nvidia/k8s/cuda-sample:nbody nbody -gpu -benchmark
```

This runs an n-body simulation benchmark on the GPU. The output will be similar to:

```
Run "nbody -benchmark [-numbodies=<numBodies>]" to measure performance.
        -fullscreen       (run n-body simulation in fullscreen mode)
        -fp64             (use double precision floating point values for simulation)
        -hostmem          (stores simulation data in host memory)
        -benchmark        (run benchmark to measure performance)
        -numbodies=<N>    (number of bodies (>= 1) to run in simulation)
        -device=<d>       (where d=0,1,2.... for the CUDA device to use)
        -numdevices=<i>   (where i=(number of CUDA devices > 0) to use for simulation)
        -compare          (compares simulation results running once on the default GPU and
once on the CPU)
        -cpu              (run n-body simulation on the CPU)
        -tipsy=<file.bin> (load a tipsy model file for simulation)

> NOTE: The CUDA Samples are not meant for performance measurements. Results may vary when GPU
Boost is enabled.

> Windowed mode
> Simulation data stored in video memory
> Single precision floating point simulation
> 1 Devices used for simulation
MapSMtoCores for SM 7.5 is undefined.  Default to use 64 Cores/SM
GPU Device 0: "GeForce RTX 2060 with Max-Q Design" with compute capability 7.5

> Compute 7.5 CUDA device: [GeForce RTX 2060 with Max-Q Design]
30720 bodies, total time for 10 iterations: 69.280 ms
= 136.219 billion interactions per second
= 2724.379 single-precision GFLOP/s at 20 flops per interaction
```

Run a real-world model: SmolLM2 with Docker Model Runner

Use Docker Model Runner to run the SmolLM2 LLM with vLLM and GPU acceleration:

```
$ docker model install-runner --backend vllm --gpu cuda
```

Check it's correctly installed:

```
$ docker model status
Docker Model Runner is running

Status:
llama.cpp: running llama.cpp version: c22473b
vllm: running vllm version: 0.11.0
```

Run the model:

```
$ docker model run ai/smollm2-vllm hi
Hello! I'm sure everything goes smoothly here. How can I assist you today?
```

□□□ #2
□ □□□ □□□ 2026-10-31 19:10:51 UTC
□ □□□ □□□ 2026-10-31 19:10:53 UTC