Running LLaMA 3–8B Instruct Inference using NVIDIA NIM on Oracle Compute Cloud@Customer and Private Cloud Appliance

A Step-by-Step Guide to Deploying and Running LLaMA 3–8B Inference with NVIDIA NIM on Oracle Compute Cloud@Customer and Private Cloud Appliance using NVIDIA L40S GPUs

Version [1.0]

Purpose statement

This document provides a step-by-step guide for deploying and running inference with the LLaMA 3–8B Instruct NVIDIA Inference Microservice (NIM) on Oracle Compute Cloud@Customer or Private Cloud Appliance. It is intended for users who have already configured their environment using the companion 'Deploying NVIDIA AI Enterprise on Oracle Compute Cloud@Customer or Private Cloud Appliance's guide.

Disclaimer

Table of contents

Introduction

Oracle Compute Cloud@Customer (C3) and Private Cloud Appliance (PCA) support NVIDIA's L40S GPU and is validated for NVIDIA AI Enterprise software, enabling the deployment of GPU-accelerated workloads including NVIDIA Inference Microservices (NIMs).

NVIDIA AI Enterprise provides certified drivers, runtime libraries, and support services needed to reliably run NVIDIA Inference Microservices in production.

NVIDIA Inference Microservice (NIM) accelerates time-to-value by simplifying deployment of optimized AI inference endpoints using pre-packaged containers. These microservices are built on NVIDIA AI Enterprise and are tuned for high-performance serving of NVIDIA-optimized models.

This technical brief provides step-by-step guidance on deploying and running inference using NVIDIA's LLaMA 3–8B Instruct NIM on Oracle Compute Cloud@Customer (C3) or Private Cloud Appliance (PCA) a core part of Oracle's Edge Cloud portfolio.

Note: This document is for informational and self-supported guidance only. For advisory services, contact your Oracle Sales representative.

Prerequisites

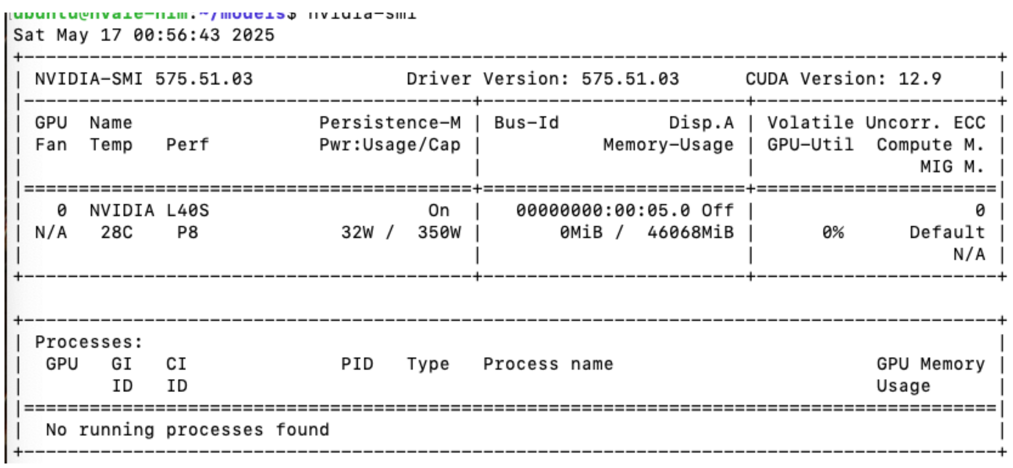
For this installation we have used NVIDIA AI Enterprise platform release 6.0, within the Compute Cloud@Customer, M3.10.3. Before beginning, ensure the following:

- Ubuntu-based VM deployed on C3 with NVIDIA L40S GPU support.
- NVIDIA AI Enterprise stack installed, including:
  - NVIDIA GPU Driver
  - CUDA Toolkit
  - NVIDIA Container Toolkit
- NGC (NVIDIA GPU Cloud) account, NGC CLI installed, and NGC API key generated.
- `jq` utility installed for parsing JSON output (e.g., `sudo apt-get install jq`)
- NVIDIA AI Enterprise deployment guide: Refer to the companion document *Deploying Nvidia AI Enterprise on Oracle Compute Cloud@Customer* or Private Cloud Appliance for environment setup steps.

Environment Validation

After deploying NVIDIA AI Enterprise, run the following commands to confirm readiness:

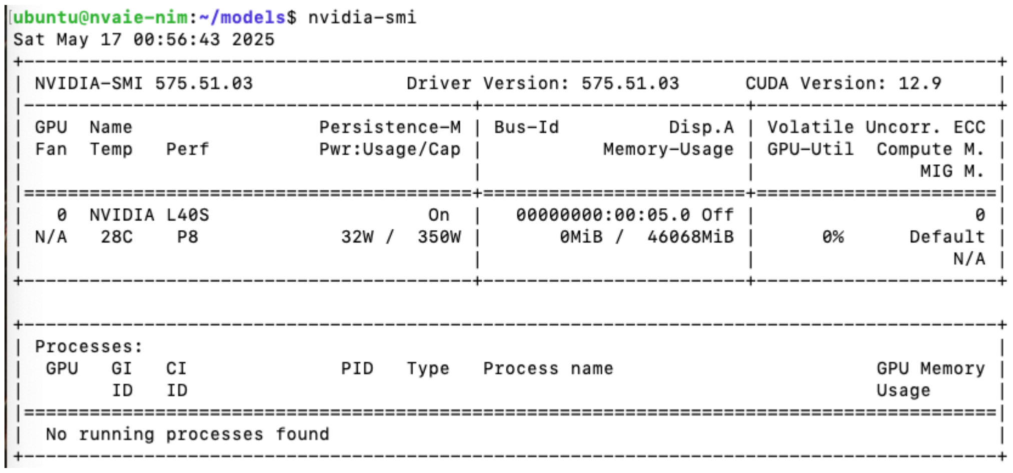
```
nvidia-smi
```



```
ubuntu@nvaie-nim:~/models$ nvidia-smi
Sat May 17 00:56:43 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 575.51.03              Driver Version: 575.51.03      CUDA Version: 12.9     |
|-----------------------------------------+------------------------+----------------------+
| 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 L40S                    On  |   00000000:00:05.0 Off |                    0 |
| N/A   28C    P8             32W /  350W |       0MiB /  46068MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI              PID   Type   Process name                        GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|  No running processes found                                                             |
+-----------------------------------------------------------------------------------------+
```

Confirms GPU driver installation and GPU visibility.

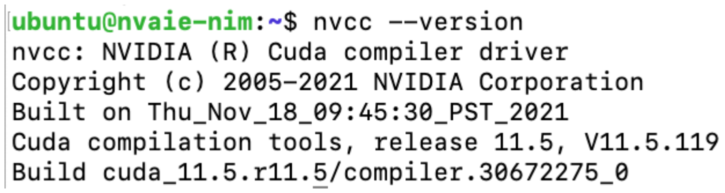
```
docker run --rm --runtime=nvidia --gpus all ubuntu nvidia-smi
```



```
ubuntu@nvaie-nim:~/models$ nvidia-smi
Sat May 17 00:56:43 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 575.51.03              Driver Version: 575.51.03      CUDA Version: 12.9     |
|-----------------------------------------+------------------------+----------------------+
| 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 L40S                    On  |   00000000:00:05.0 Off |                    0 |
| N/A   28C    P8             32W /  350W |       0MiB /  46068MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI              PID   Type   Process name                        GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|  No running processes found                                                             |
+-----------------------------------------------------------------------------------------+
```

Validates Docker’s GPU access through the NVIDIA Container Toolkit.

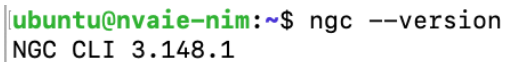
```
nvcc --version
```



```
ubuntu@nvaie-nim:~$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2021 NVIDIA Corporation
Built on Thu_Nov_18_09:45:30_PST_2021
Cuda compilation tools, release 11.5, V11.5.119
Build cuda_11.5.r11.5/compiler.30672275_0
```

Verifies that the CUDA toolkit is correctly installed.

```
ngc --version
```



```
ubuntu@nvaie-nim:~$ ngc --version
NGC CLI 3.148.1
```

Confirms NGC CLI is installed.

```
Echo $NGC_API_KEY
```

Ensures API key is set.

```
echo "$NGC_API_KEY" | docker login nvcr.io --username '$oauthtoken' --password-stdin
```

Logs in to NVIDIA container registry.

```
ngc registry image list --format_type=csv nvcr.io/nim/* | column -t -s,
```

Lists all available NIMs from NGC catalog.

Setup Script for Inference

Prepare a setup script (eg. LLaMA3-8b-Instruct) to streamline the inferencing workflow:

# Export API key

```
export NGC_API_KEY="<your-api-key>"
```

# Persist the key across sessions

```
echo "export NGC_API_KEY=\"$NGC_API_KEY\"" >> ~/.bashrc
```

# Docker login

```
echo "$NGC_API_KEY" | docker login nvcr.io --username '$oauthtoken' --password-stdin
```

# Define image

```
export CONTAINER_NAME=Llama3-8B-Instruct
export Repository=nim/meta/llama3-8b-instruct
export IMG_NAME="nvcr.io/${Repository}:latest"
```

# Setup model cache

```
export LOCAL_NIM_CACHE=~/.cache/nim
mkdir -p "$LOCAL_NIM_CACHE"
chmod -R a+w "$LOCAL_NIM_CACHE"
```

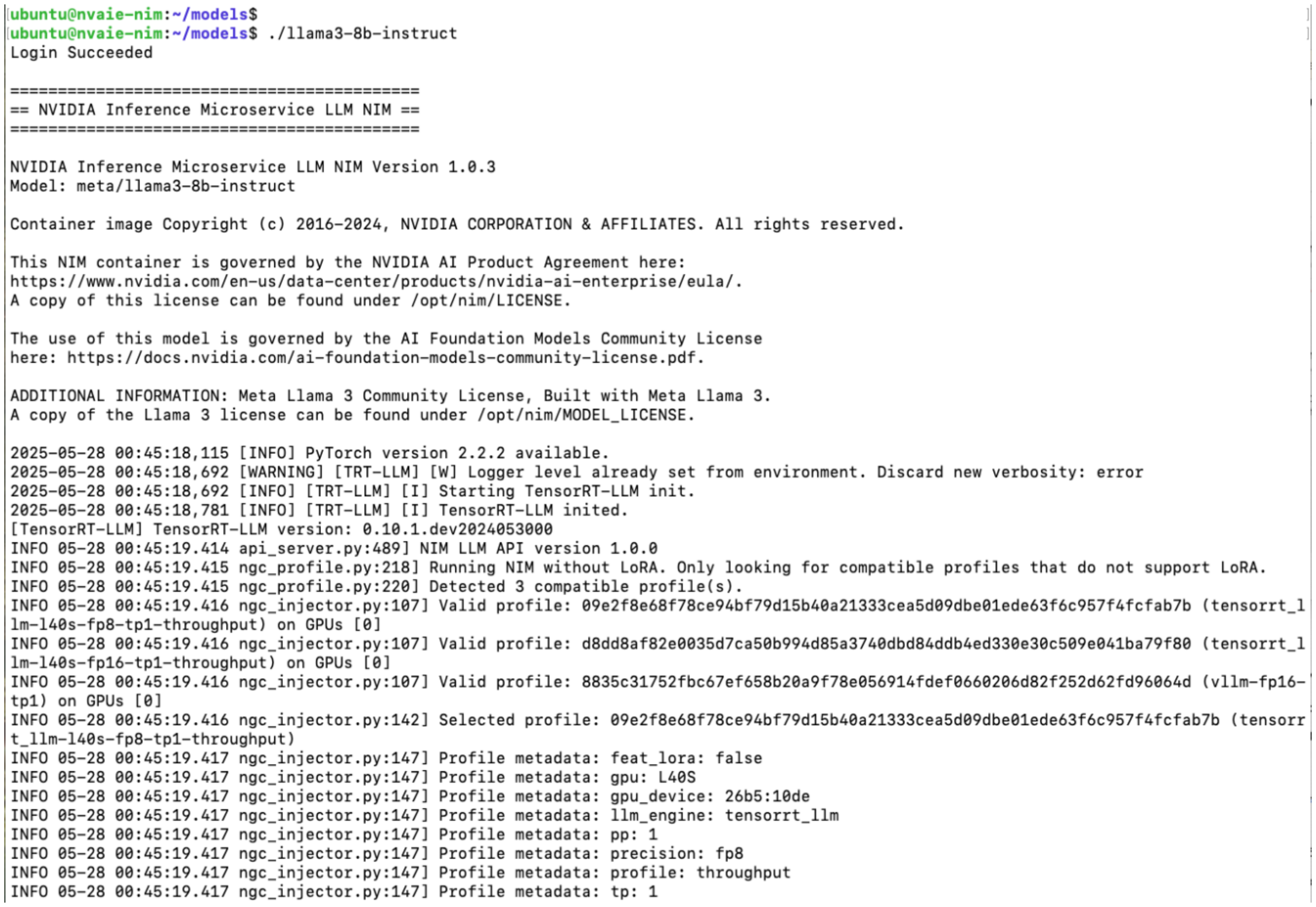
# Run NIM container

```
sudo docker run --rm -it \
  --runtime=nvidia \
  --gpus all \
  --shm-size=16gb \
  -e "NGC_API_KEY=$NGC_API_KEY" \
  -v "$LOCAL_NIM_CACHE:/opt/nim/.cache" \
  -u $(id -u):$(id -g) \
  --network=host \
  --name "$CONTAINER_NAME" \
  "$IMG_NAME"
```

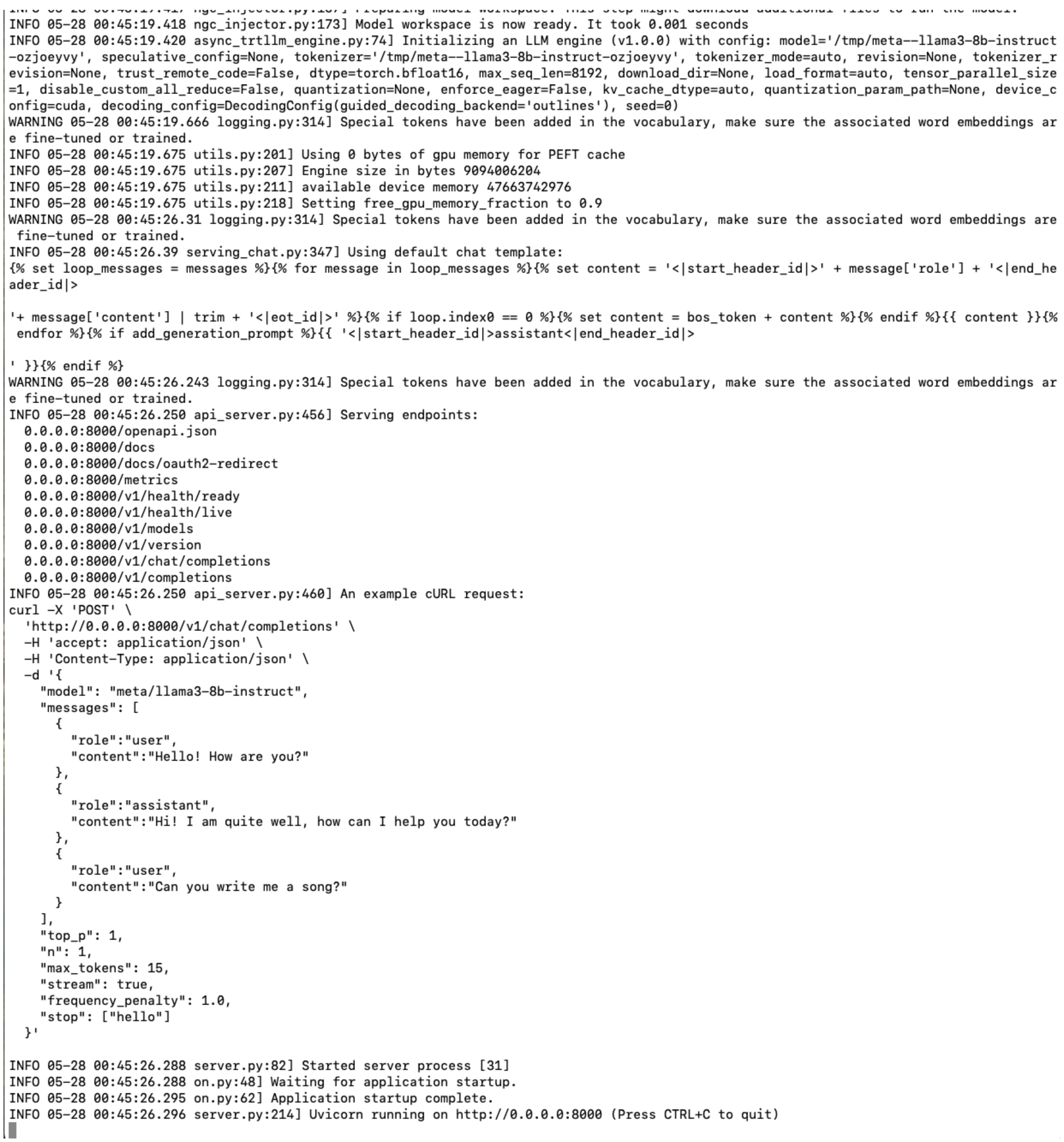
#-p 8000:8000 maps the container's inference endpoint to the host machine

Note: `-p 8000:8000` maps the container's inference endpoint to the host machine. In this setup, since the host port was not the same as the container port, we used `--network=host` instead. This mode is helpful in environments with restrictive Docker networking, firewall rules, or potential port conflicts. It exposes all container ports directly to the host, simplifying connectivity.

Run the script



```
ubuntu@nvaie-nim:~/models$
ubuntu@nvaie-nim:~/models$ ./llama3-8b-instruct
Login Succeeded

==========================================
== NVIDIA Inference Microservice LLM NIM ==
==========================================

NVIDIA Inference Microservice LLM NIM Version 1.0.3
Model: meta/llama3-8b-instruct

Container image Copyright (c) 2016-2024, NVIDIA CORPORATION & AFFILIATES. All rights reserved.

This NIM container is governed by the NVIDIA AI Product Agreement here:
https://www.nvidia.com/en-us/data-center/products/nvidia-ai-enterprise/eula/.
A copy of this license can be found under /opt/nim/LICENSE.

The use of this model is governed by the AI Foundation Models Community License
here: https://docs.nvidia.com/ai-foundation-models-community-license.pdf.

ADDITIONAL INFORMATION: Meta Llama 3 Community License, Built with Meta Llama 3.
A copy of the Llama 3 license can be found under /opt/nim/MODEL_LICENSE.

2025-05-28 00:45:18,115 [INFO] PyTorch version 2.2.2 available.
2025-05-28 00:45:18,692 [WARNING] [TRT-LLM] [W] Logger level already set from environment. Discard new verbosity: error
2025-05-28 00:45:18,692 [INFO] [TRT-LLM] [I] Starting TensorRT-LLM init.
2025-05-28 00:45:18,781 [INFO] [TRT-LLM] [I] TensorRT-LLM inited.
[TensorRT-LLM] TensorRT-LLM version: 0.10.1.dev2024053000
INFO 05-28 00:45:19.414 api_server.py:489] NIM LLM API version 1.0.0
INFO 05-28 00:45:19.415 ngc_profile.py:218] Running NIM without LoRA. Only looking for compatible profiles that do not support LoRA.
INFO 05-28 00:45:19.415 ngc_profile.py:220] Detected 3 compatible profile(s).
INFO 05-28 00:45:19.416 ngc_injector.py:107] Valid profile: 09e2f8e68f78ce94bf79d15b40a21333cea5d09dbe01ede63f6c957f4fcfab7b (tensorrt_l
lm-l40s-fp8-tp1-throughput) on GPUs [0]
INFO 05-28 00:45:19.416 ngc_injector.py:107] Valid profile: d8dd8af82e0035d7ca50b994d85a3740dbd84ddb4ed330e30c509e041ba79f80 (tensorrt_l
lm-l40s-fp16-tp1-throughput) on GPUs [0]
INFO 05-28 00:45:19.416 ngc_injector.py:107] Valid profile: 8835c31752fbc67ef658b20a9f78e056914fdef0660206d82f252d62fd96064d (vllm-fp16-
tp1) on GPUs [0]
INFO 05-28 00:45:19.416 ngc_injector.py:142] Selected profile: 09e2f8e68f78ce94bf79d15b40a21333cea5d09dbe01ede63f6c957f4fcfab7b (tensorr
t_llm-l40s-fp8-tp1-throughput)
INFO 05-28 00:45:19.417 ngc_injector.py:147] Profile metadata: feat_lora: false
INFO 05-28 00:45:19.417 ngc_injector.py:147] Profile metadata: gpu: L40S
INFO 05-28 00:45:19.417 ngc_injector.py:147] Profile metadata: gpu_device: 26b5:10de
INFO 05-28 00:45:19.417 ngc_injector.py:147] Profile metadata: llm_engine: tensorrt_llm
INFO 05-28 00:45:19.417 ngc_injector.py:147] Profile metadata: pp: 1
INFO 05-28 00:45:19.417 ngc_injector.py:147] Profile metadata: precision: fp8
INFO 05-28 00:45:19.417 ngc_injector.py:147] Profile metadata: profile: throughput
INFO 05-28 00:45:19.417 ngc_injector.py:147] Profile metadata: tp: 1
```



```
INFO 05-28 00:45:19.417 ngc_injector.py:167] Preparing model workspace. This step might download additional files to run the model.
INFO 05-28 00:45:19.418 ngc_injector.py:173] Model workspace is now ready. It took 0.001 seconds
INFO 05-28 00:45:19.420 async_trtllm_engine.py:74] Initializing an LLM engine (v1.0.0) with config: model='/tmp/meta--llama3-8b-instruct
-ozjoeyvy', speculative_config=None, tokenizer='/tmp/meta--llama3-8b-instruct-ozjoeyvy', tokenizer_mode=auto, revision=None, tokenizer_r
evision=None, trust_remote_code=False, dtype=torch.bfloat16, max_seq_len=8192, download_dir=None, load_format=auto, tensor_parallel_size
=1, disable_custom_all_reduce=False, quantization=None, enforce_eager=False, kv_cache_dtype=auto, quantization_param_path=None, device_c
onfig=cuda, decoding_config=DecodingConfig(guided_decoding_backend='outlines'), seed=0)
WARNING 05-28 00:45:19.666 logging.py:314] Special tokens have been added in the vocabulary, make sure the associated word embeddings ar
e fine-tuned or trained.
INFO 05-28 00:45:19.675 utils.py:201] Using 0 bytes of gpu memory for PEFT cache
INFO 05-28 00:45:19.675 utils.py:207] Engine size in bytes 9094006204
INFO 05-28 00:45:19.675 utils.py:211] available device memory 47663742976
INFO 05-28 00:45:19.675 utils.py:218] Setting free_gpu_memory_fraction to 0.9
WARNING 05-28 00:45:26.31 logging.py:314] Special tokens have been added in the vocabulary, make sure the associated word embeddings are
 fine-tuned or trained.
INFO 05-28 00:45:26.39 serving_chat.py:347] Using default chat template:
{% set loop_messages = messages %}{% for message in loop_messages %}{% set content = '<|start_header_id|>' + message['role'] + '<|end_he
ader_id|>

'+ message['content'] | trim + '<|eot_id|>' %}{% if loop.index0 == 0 %}{% set content = bos_token + content %}{% endif %}{{ content }}{%
 endfor %}{% if add_generation_prompt %}{{ '<|start_header_id|>assistant<|end_header_id|>

' }}{% endif %}
WARNING 05-28 00:45:26.243 logging.py:314] Special tokens have been added in the vocabulary, make sure the associated word embeddings ar
e fine-tuned or trained.
INFO 05-28 00:45:26.250 api_server.py:456] Serving endpoints:
  0.0.0.0:8000/openapi.json
  0.0.0.0:8000/docs
  0.0.0.0:8000/docs/oauth2-redirect
  0.0.0.0:8000/metrics
  0.0.0.0:8000/v1/health/ready
  0.0.0.0:8000/v1/health/live
  0.0.0.0:8000/v1/models
  0.0.0.0:8000/v1/version
  0.0.0.0:8000/v1/chat/completions
  0.0.0.0:8000/v1/completions
INFO 05-28 00:45:26.250 api_server.py:460] An example cURL request:
curl -X 'POST' \
  'http://0.0.0.0:8000/v1/chat/completions' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "model": "meta/llama3-8b-instruct",
    "messages": [
      {
        "role":"user",
        "content":"Hello! How are you?"
      },
      {
        "role":"assistant",
        "content":"Hi! I am quite well, how can I help you today?"
      },
      {
        "role":"user",
        "content":"Can you write me a song?"
      }
    ],
    "top_p": 1,
    "n": 1,
    "max_tokens": 15,
    "stream": true,
    "frequency_penalty": 1.0,
    "stop": ["hello"]
  }'

INFO 05-28 00:45:26.288 server.py:82] Started server process [31]
INFO 05-28 00:45:26.288 on.py:48] Waiting for application startup.
INFO 05-28 00:45:26.295 on.py:62] Application startup complete.
INFO 05-28 00:45:26.296 server.py:214] Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

Verify NIM Service

Once the container starts, verify the service health from another terminal:

```
curl localhost:8000/v1/health/ready
```

A successful response confirms that the NIM service is running and listening for requests. Ensure the container has fully started and is serving before sending the inference request.

Send Inference Request

Now we send an inference request to our llma3-8b mode, asking the model via the curl command to compose a poem about redwood forest:

```
curl -s -X POST http://0.0.0.0:8000/v1/completions \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "model": "meta/llama3-8b-instruct",
    "prompt": "Write a 2 para poem about california's redwood forests. Format as poem",
    "max_tokens": 80,
    "temperature": 0.9
  }' | jq -r '.choices[0].text'
```

Sample output:

```
Silent sentinels of ancient earth,
Towering canopies of green rebirth,
Majestic redwoods stand, a sentinel true,
Guarding secrets of a forest anew.

In twilight hush, where cypress whispers low,
The redwoods' ancient voices whisper slow,
Timeless wisdom shared, of sun-kissed days,
Echoes of the past, in whispers that sway
```

Additional Resources

- Deployment NVIDIA AI Enterprise on Oracle Compute Cloud@Customer
- NIM Demo: Running Inferencing Using NVIDIA NIM using Oracle Compute Cloud@Customer

Acknowledgements

- Sheetal Sabharwal, Principal Product Manager, Oracle Edge Cloud
- Salman Ashfaq, Master Principal Sales Consultant, Oracle Solution Center
- Michael Reed, Solution Specialist Director, Oracle Solution Center

Connect with us

Call +**1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com facebook.com/oracle twitter.com/oracle

Author: Sheetal Sabharwal, Anderson Souza