

Sizing LLM Inference Systems

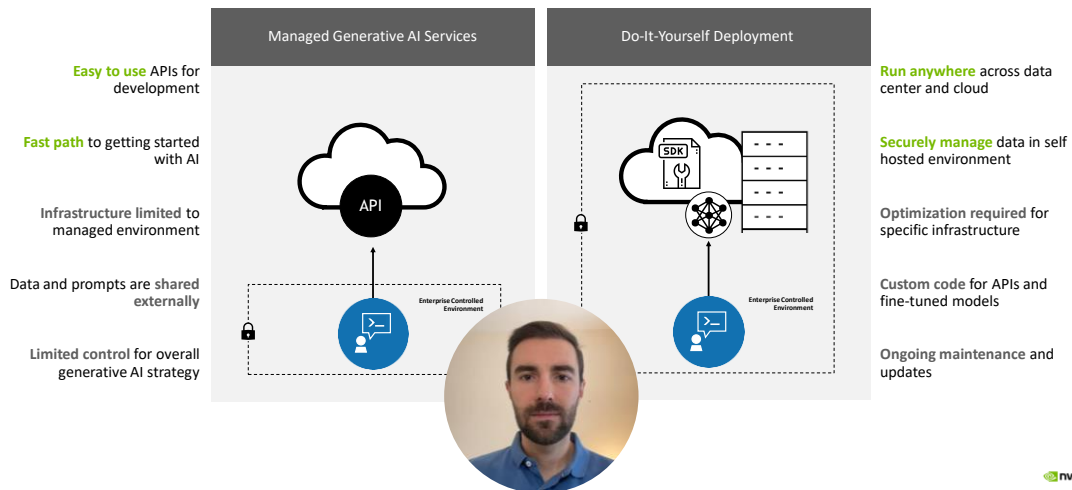
First Contact with NIMs



Hi again, and welcome to this first notebook where you will experiment with NIMs, the NVIDIA inference microservices. Let me explain to you some basic concepts about NIM before you work on the notebook.

Enterprises Face Challenges Experimenting with Generative AI

Organizations must choose between ease of use and control



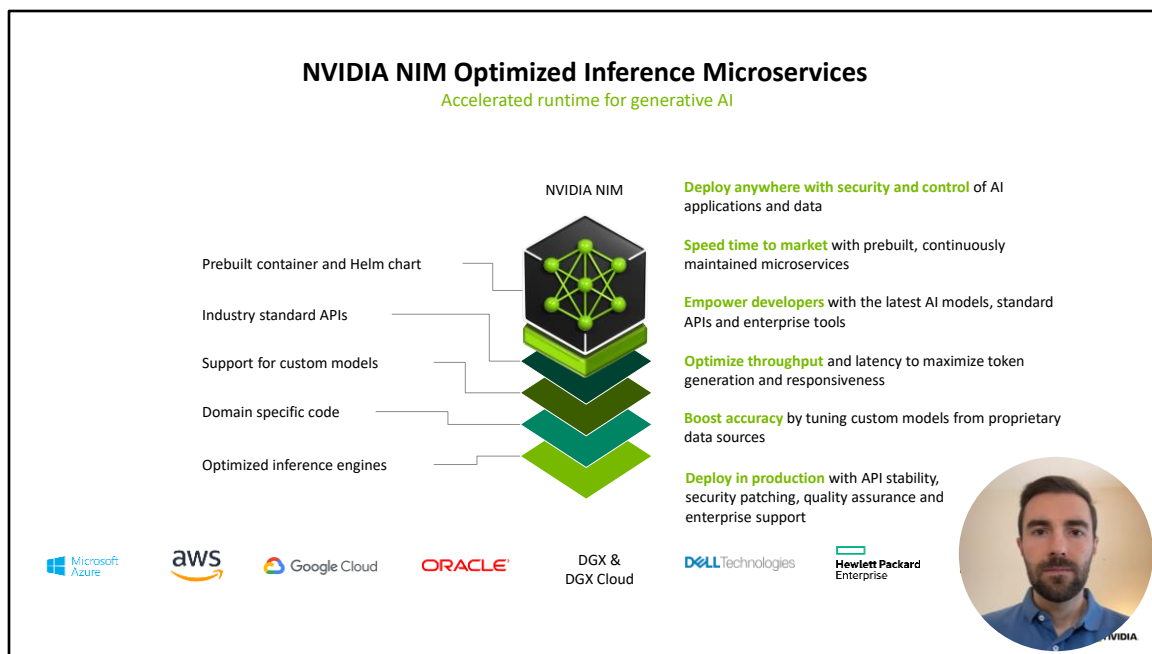
Many of the companies I talk to have already begun the process of building AI applications.

On one side, we have Managed Generative AI Services. They have simple and user-friendly APIs and are great to get started.

However, these benefits come with certain trade-offs, like limited control over the infrastructure, data usage or generative AI strategy.

On the flip side, we have Open-Source Deployments. Think about the TensorRT-LLM and Triton libraries for developers, which offer a do-it-yourself approach for inference. You have a lot of control about where to deploy and security.

However, this increased control also means that you need to optimize the inference workload yourself, writing for example customer code or fine-tuning models.



NVIDIA NIM offers the best of both worlds and it is a simple way to deploy AI models. It offers many advantages, and in this slide we list some of them.

For instance, it offers compatibility with standard APIs. If you have built your application using a different Generative AI service, you can just swap the endpoint url to point to NIM.





NIMs are also optimized to run efficiently for inference. In this DLI, you are going to learn to optimize inference workloads for your particular latency and throughput requirements. However, with NIMs we give you those optimizations out-of-the-box, so that you can quickly deploy your LLM.

We do have many NIMs available for models in language, vision or biology, so do check them out. With our developer program, you can access them for free. You can also leverage NIMs for production via NVIDIA's AI Enterprise license.

In this notebook, you will have the chance of testing a NIM yourself.

NVIDIA NIM is the Fastest Path to AI Inference

Reduces engineering resources required to deploy optimized, accelerated models

	NVIDIA NIM	Do It Yourself
Deployment Time	5 minutes	1 week +
API Standardization	Industry standard protocol OpenAI for LLMs, Google Translate for Speech	Implement the API layer for each domain and model family according to industry standard specifications
Optimized Engines	Pre-built engines for NVIDIA and community models    	Build your own engine and manually customize for workload and hardware specific requirements
Pre and Post Processing Pipelines	Pre-built with optimized pipeline engines to handle pre/post processing (tokenization)	Implement custom logic
Model Server Deployment	Automated	Manual setup and configuration
Customization	LoRA is supported, more planned	Create custom logic
Container Validation	Extensive workload specific QA support matrix validation	No validation
Enterprise Support	Delivered with NVIDIA AI Enterprise Security and CVE scanning/patching and tech support	Self supported



Here we show some of the advantages of NIM with respect to a do it yourself approach. The deployment time is quite significant, from minutes in NIM to several days in the do it yourself.

NIMs also offer API standardisation, optimized engines built with tensorrt-llm, and several other advantages listed here.

Do take some time to read them out.

NVIDIA NIM for LLM

Deploy an Optimized NIM with a Single Command

- **NVIDIA NIM for LLM offers per-model containers**

- When a model container is launched, it:
 - Detects the hardware it is running on
 - Mounts the cache for model and asset data
 - Selects the most optimal model given the hardware
 - Downloads the optimized model file from NGC
 - Loads the model file and starts serving

Model-specific Config Image
(llama3-8b-instruct)

Base Container Image
(NIM for LLMs)

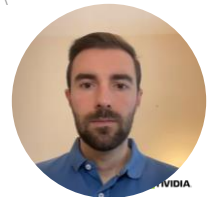
```
export NGC_API_KEY=<value>
echo "$NGC_API_KEY"
docker login nvcr.io --username 'Soauthtoken' --password-stdin

# Choose a container name for bookkeeping
export CONTAINER_NAME=llama3-8b-instruct

# Choose a LLM NIM Image from NGC
export IMG_NAME="nvcr.io/nim/meta/{CONTAINER_NAME}:1.0.0"

# Choose a path on your system to cache the downloaded models
export LOCAL_NIM_CACHE=~/.cache/nim
mkdir -p "$LOCAL_NIM_CACHE"

# Start the LLM NIM
docker run -it --rm --name=$CONTAINER_NAME \
  --runtime=nvidia \
  --gpus all \
  --shm-size=16GB \
  -e NGC_API_KEY \
  -v "$LOCAL_NIM_CACHE:/opt/nim/.cache" \
  -u $(id -u) \
  -p 8000:8000 \
  $IMG_NAME
```



A NIM is basically a container exposing an API. Each NIM container is specific to the specific LLM it will serve.

There is a base container image across all LLM NIMs, and a model configuration layer with model and asset data. The design point here is to make deploying multiple NIMs as fast as possible by reusing the base container image and only newly downloading different lightweight model config layers for the container.

NIMs are deployed by launching the NIM container. Upon launch the NIM:

- Detects the underlying hardware
- Mounts the cache for the lightweight model config layer of model and asset data
- Selects the most optimal version of the model based on the hardware detected
- Downloads the optimized model from NGC
- And loads the model and starts serving it through a REST API endpoint

Llama-3-8B-Instruct Optimization for NVIDIA GPUs with LoRA Support Option



The result is just delicious.

Notebook Objectives

Notebook 0

1. Get familiar with NIM deployment and usage
2. Call the NIM endpoint with curl and python to generate some text
3. Consider End-to-End Latency (E2E) versus Time-to-First-Token Latency (TTFT)



After this introduction to NIM, you are ready to tackle this first notebook!

In it you will get familiar with NIMs, you will call the NIM endpoint with curl and python to generate some text, and you will measure the end to end latency versus time to first token.

Now it's your chance to apply all you've learned about NIMs.

