
# Efficient Large Language Model Customization

# Agenda

Prompt Engineering

Retrieval Augmented Generation (RAG)

Parameter-Efficient Fine-Tuning Essentials

- P-Tuning
- Low Rank Adaptation (LoRA)

LLM Serving and Deployment

# Agenda

Prompt Engineering

Retrieval Augmented Generation (RAG)

Parameter-Efficient Fine-Tuning Essentials

- P-Tuning
- Low Rank Adaptation (LoRA)

LLM Serving and Deployment

# Prompt Engineering

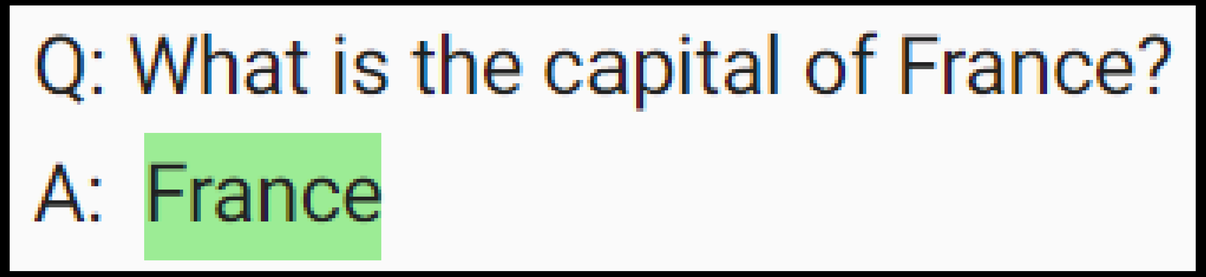
- The term prompt engineering refers to the process of carefully designing the prompts to generate a specific output.
- The following examples discuss three different strategies:
  - Zero-shot prompts
  - Few-shot prompts
  - Chain-of-thought prompts

# Prompt Engineering

## Zero-Shot Prompt

- Zero-shot means prompting the model without any example of expected behavior from the model.

- In the example, the answer is incorrect, as Paris is the capital. Judging from the answer, the model may not understand the use of the word “capital” in this context.



# Prompt Engineering

## Few-Shot Prompt

- A simple way to overcome this issue is to give some examples in the prompt.
- This type of prompt is referred to as a *few-shot prompt*.
- You provide a few examples before posing the actual question.
- A few-shot prompt enables the model to learn without training.

Q: What is the capital of Spain?

A: Madrid

Q: What is the capital of Italy?

A: Rome

Q: What is the capital of France?

A: Paris

# Prompt Engineering

## Few-Shot Chain of Thought Prompt

- How to get the model to answer questions logically (teach reasoning)?
- In the example, the correct answer is “4 blue golf balls”.
- To help develop reasoning, use a prompting method called chain-of-thought prompting.
- To do this, we provide some few-shot examples, where the reasoning process is explained.
- When the LLM answers the prompt, it shows its reasoning process as well.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: There are 8 blue golf balls.

Q: A juggler can juggle 4 balls. Half of the balls are golf balls, and half of the golf balls are red. How many red golf balls are there?
A: Step1: There are 4 balls. Step 2: There are 2 golf balls. Step3: There is 1 red golf ball. So there is 1 red golf ball.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: Step1: There are 16 balls. Step 2: There are 8 golf balls. Step3: There is 4 blue golf balls. So there is 4 blue golf balls.

# Prompt Engineering

## Few-Shot Chain of Thought Prompt

- You can also do a Zero-shot Chain of Thought prompt as shown in the example.
- Such prompts usually include phrases such as, “Let’s think about this logically.”

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there? Let's think about this logically.

1. There are 16 balls.
2. Half of the balls are golf balls.
3. Half of the golf balls are blue.

Now we can write an equation.

There are 8 golf balls.

There are 4 blue golf balls.

So, the answer is 4.

# Prompt Engineering

## Shortcomings

- A small number of examples can be used, limiting the level of control.
- The examples must be pre-appended, which affects the token budget

# Agenda

Prompt Engineering

Retrieval Augmented Generation (RAG)

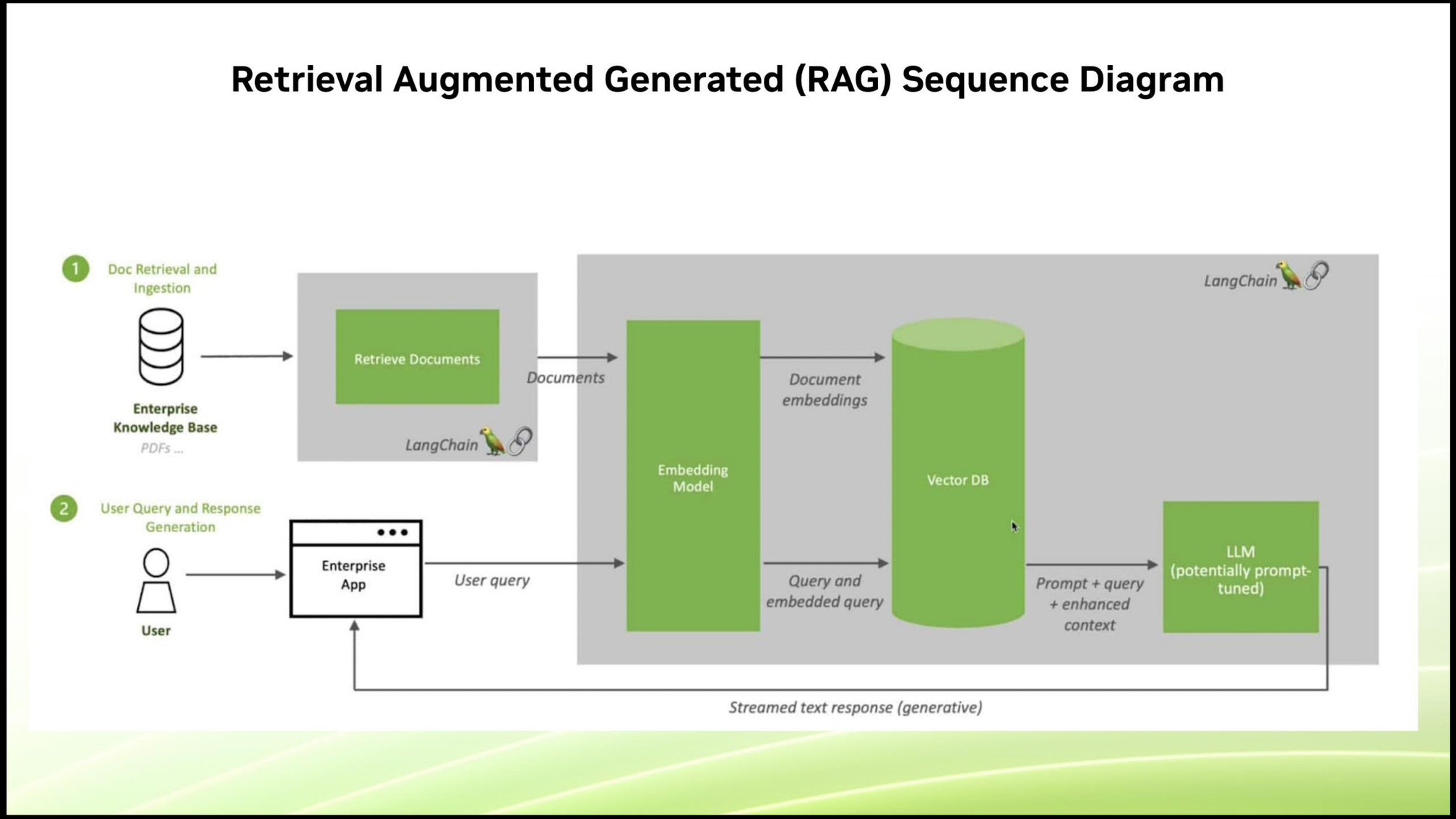
Parameter-Efficient Fine-Tuning Essentials

- P-Tuning
- Low Rank Adaptation (LoRA)

LLM Serving and Deployment

# Retrieval Augmented Generation

RAG



# Agenda

Prompt Engineering

Retrieval Augmented Generation (RAG)

Parameter-Efficient Fine-Tuning Essentials

- P-Tuning
- Low Rank Adaptation (LoRA)

LLM Serving and Deployment

# Parameter Efficient Fine Tuning

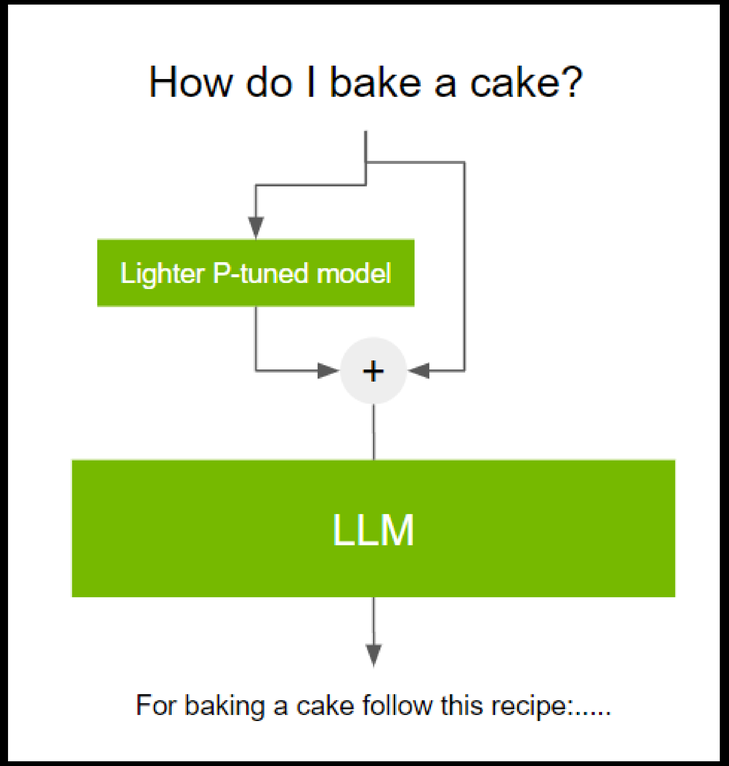
## PEFT

- To overcome limitations of Prompt Engineering:
  - Transfer learning is an obvious candidate: start with a base model and use the use case–specific data to fine-tune the model.
  - This approach works well when dealing with regular models, but fine-tuning a model with 530B parameters (about 5,300x larger than a BERT model) consumes considerable time and resources.


NVIDIA
P-Tuning

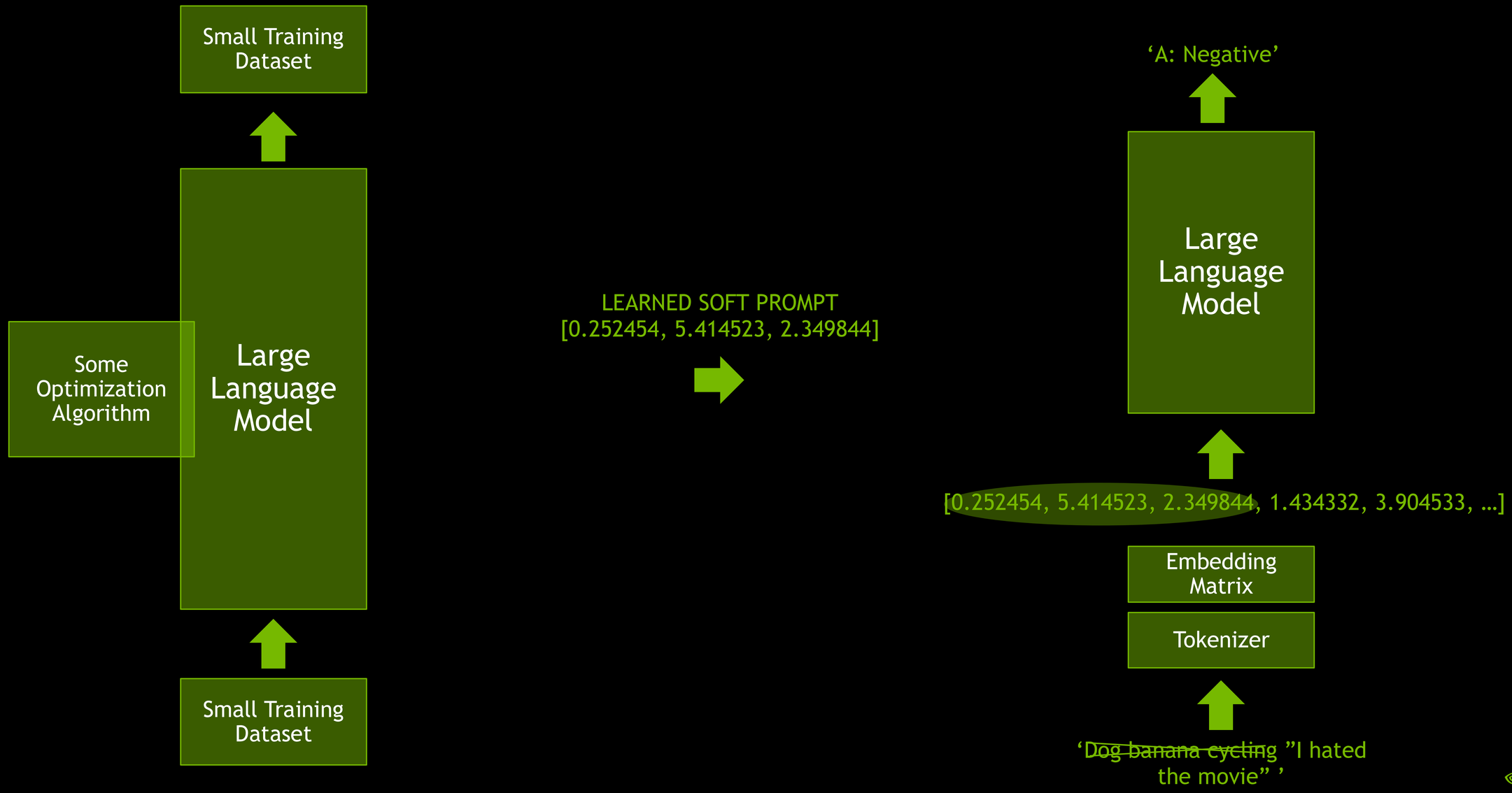
# P-Tuning

- P-tuning, or *prompt tuning*, is a parameter-efficient tuning technique that involves using a small trainable model before using the LLM.
- The small model is used to encode the text prompt and generate task-specific virtual tokens.
- Considerably fewer resources are required (compared to fine-tuning the LLM) to customize the model pipeline to achieve the desired results.
- The time required to tune a smaller model is much less.
- Models p-tuned on different tasks can be saved, without the need for large amounts of memory.

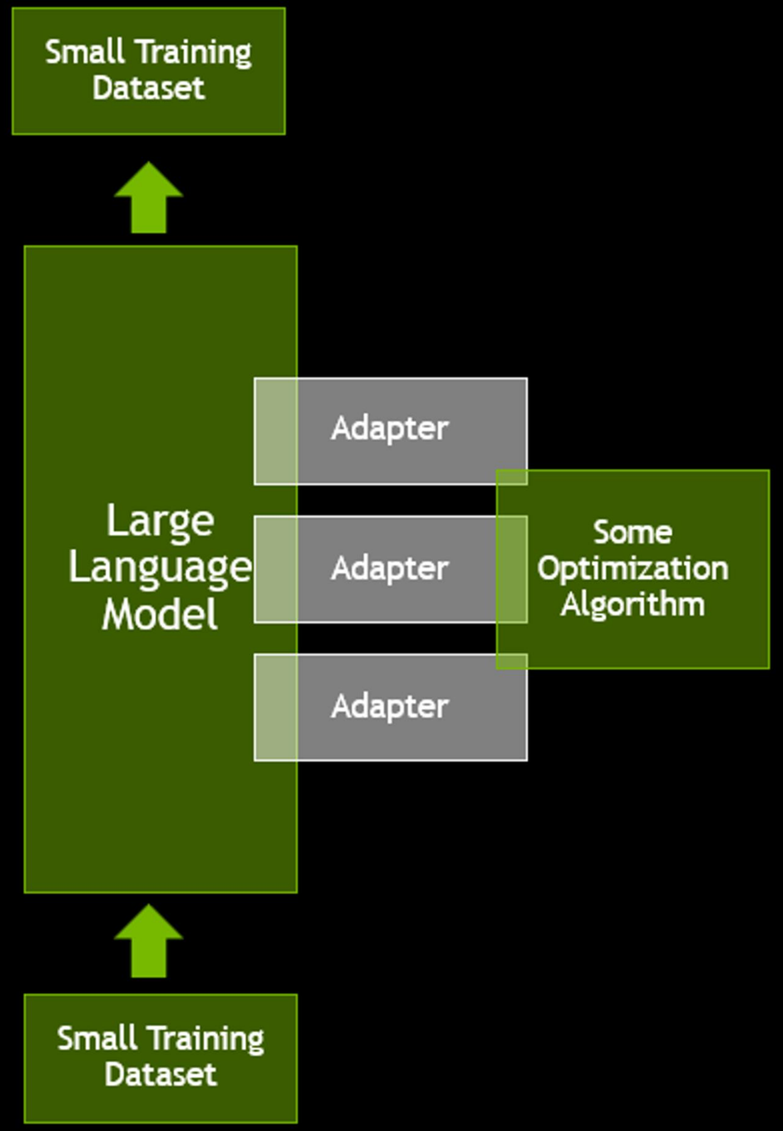


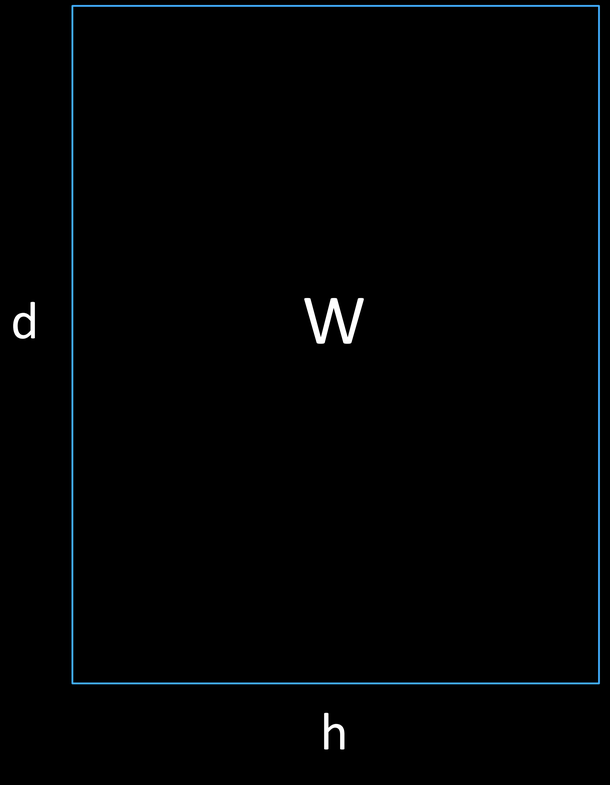
# Prompt Learning

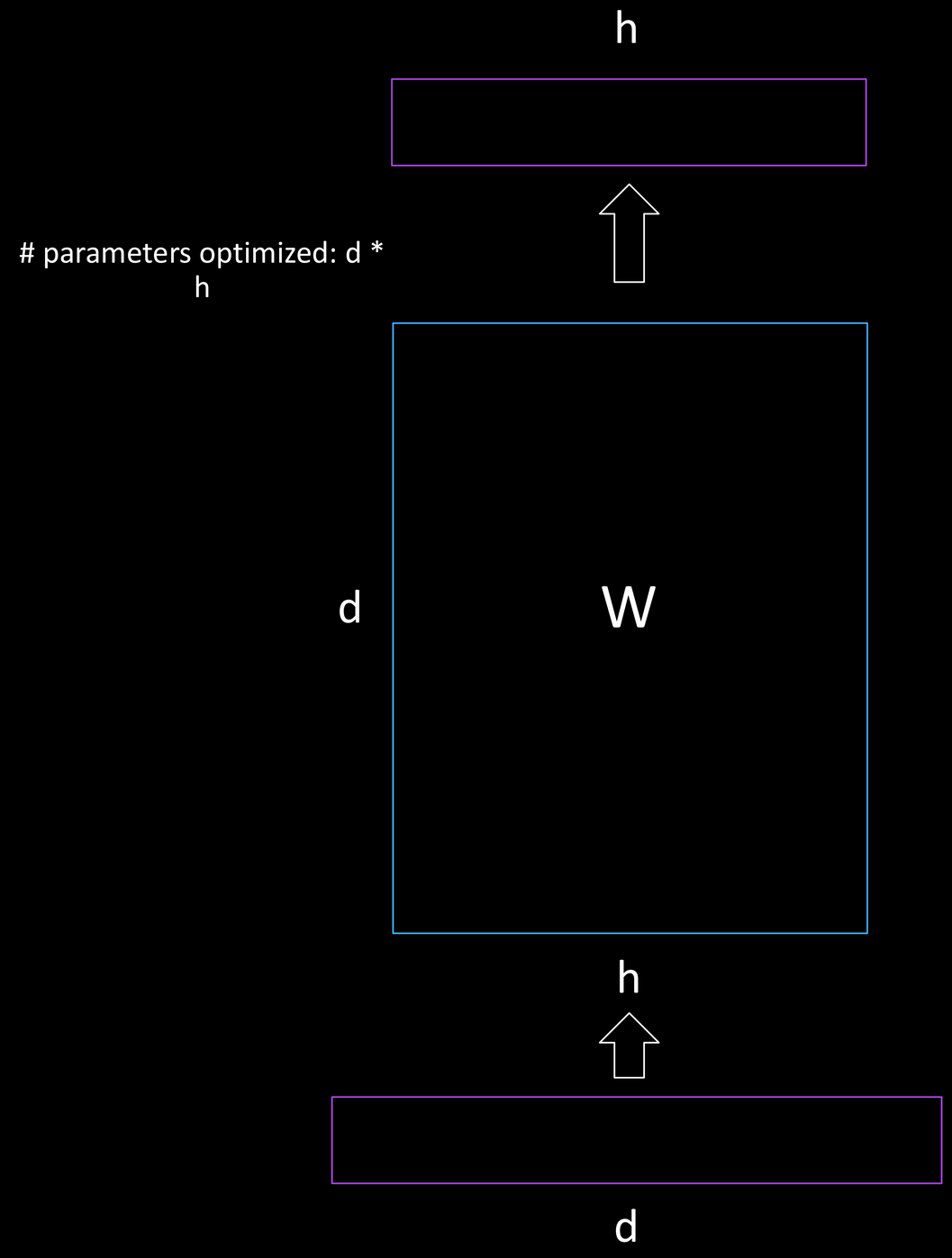
## Soft / Continuous Prompts

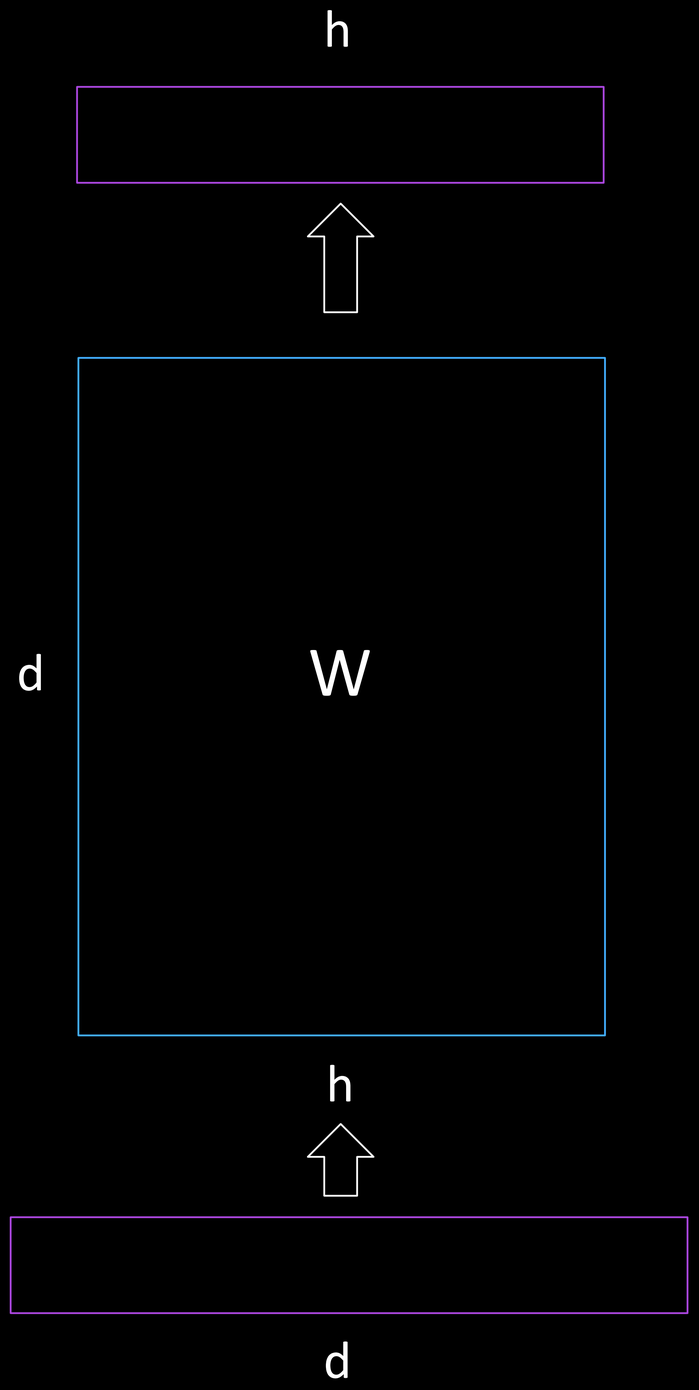



NVIDIA
LoRA

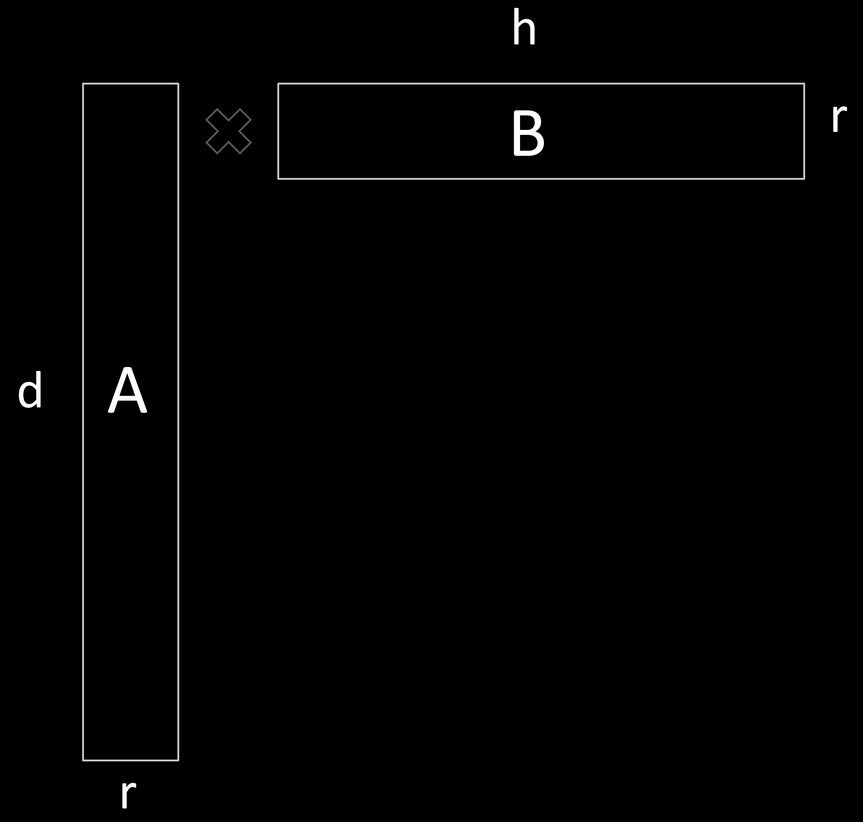

Small Training Dataset
Large Language Model
Adapter
Adapter
Adapter
Some Optimization Algorithm
Small Training Dataset

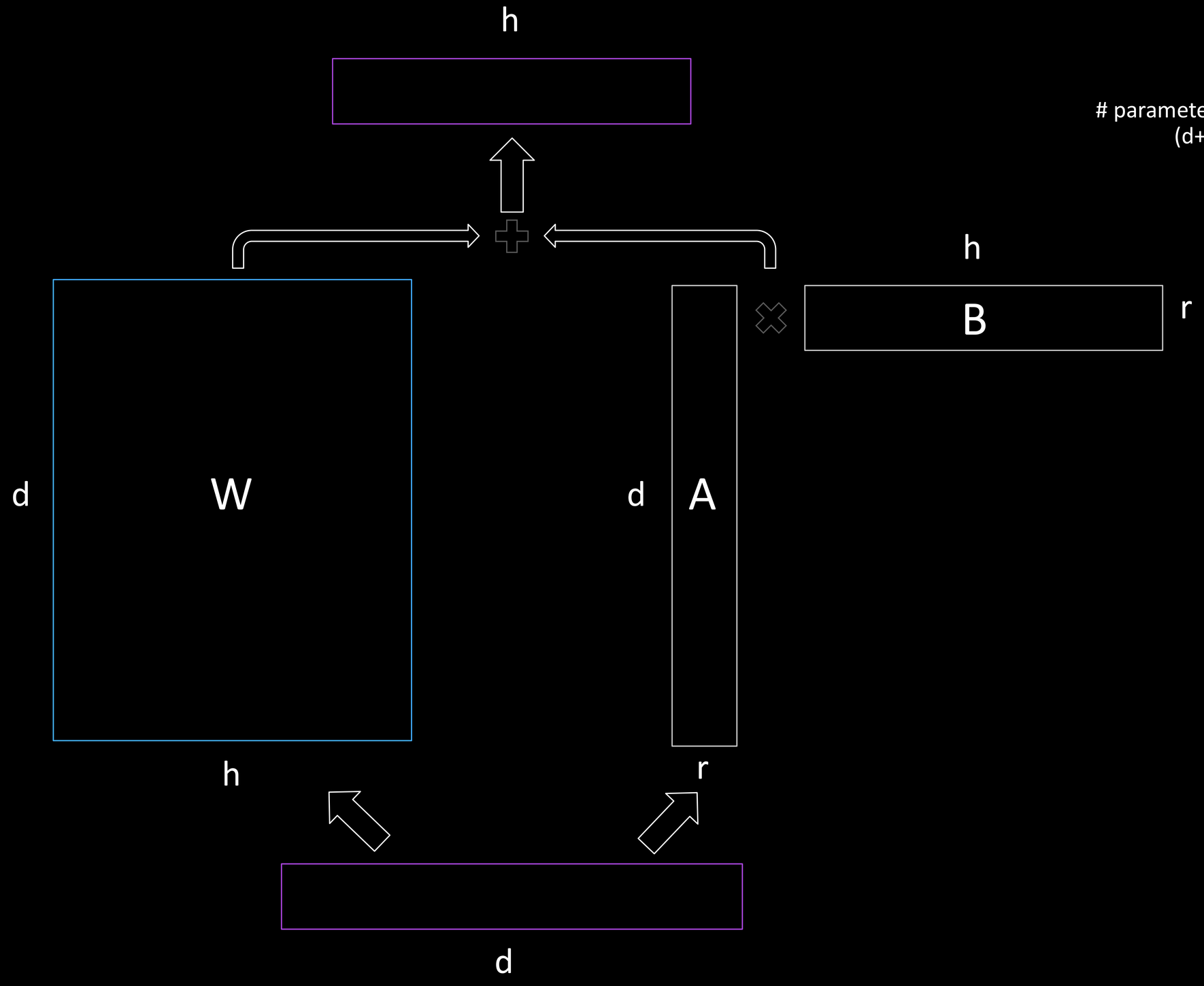

d
W
h


h
# parameters optimized: d * h
d
W
h
d




# parameters optimized:
(d+h) * r




h
# parameters optimized:
(d+h) * r
h
B
r
d
W
d
A
h
r
d

# Agenda

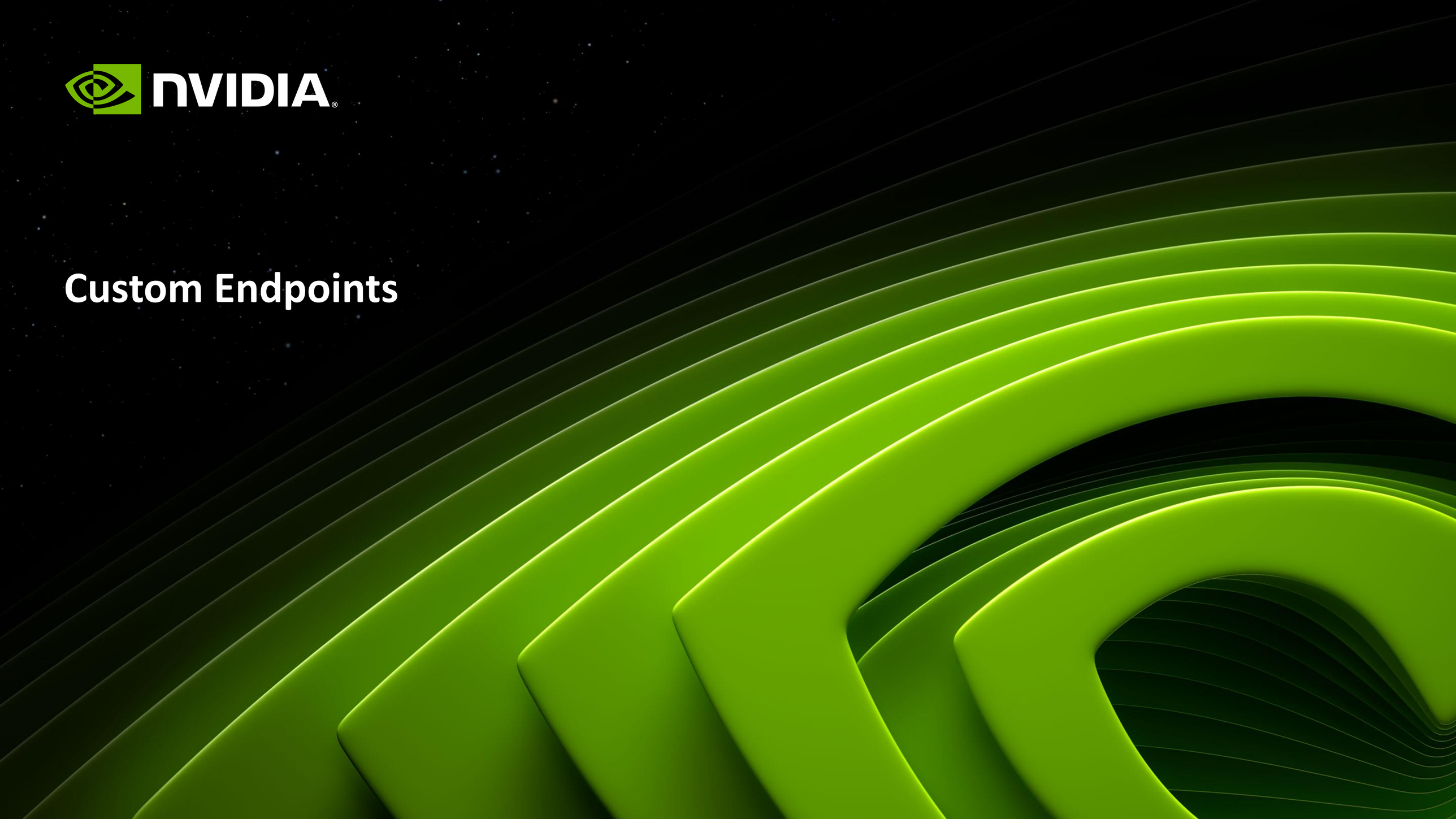
Prompt Engineering

Retrieval Augmented Generation (RAG)

Parameter-Efficient Fine-Tuning Essentials

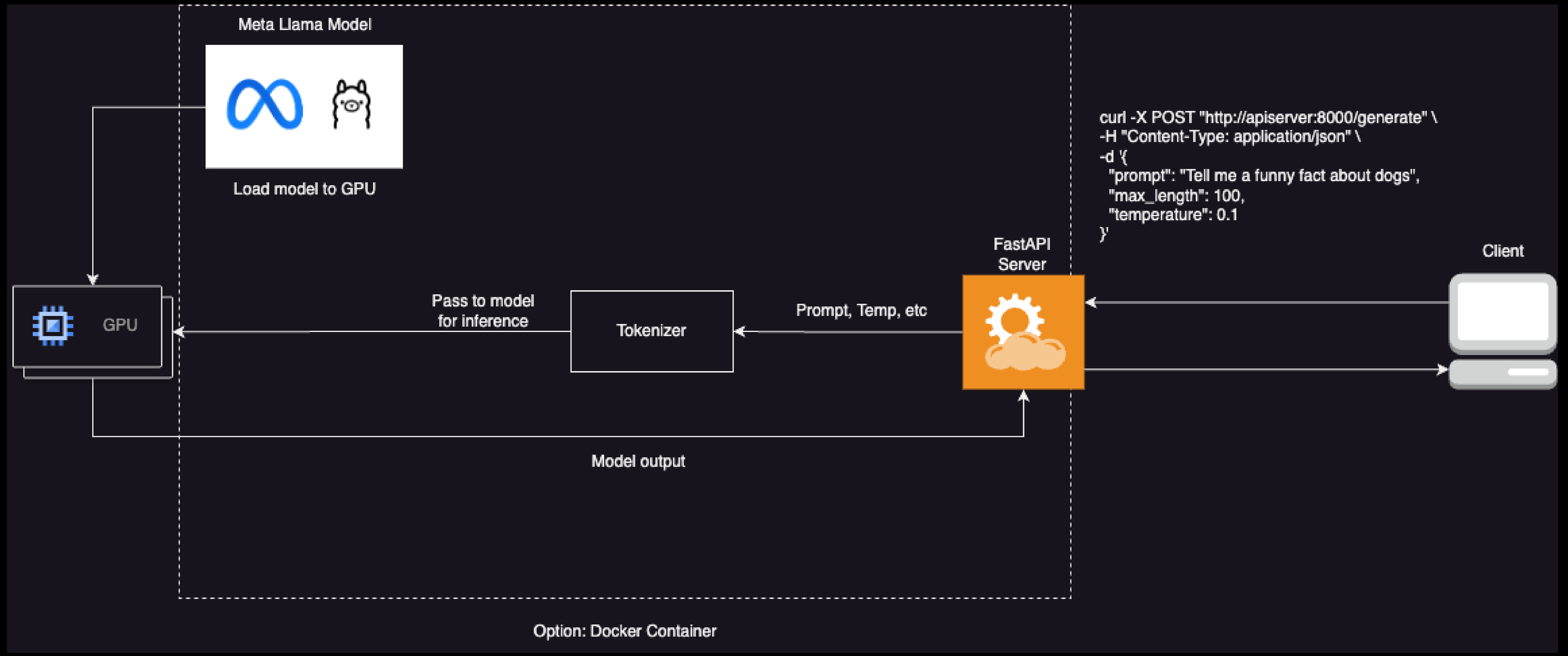
- P-Tuning
- Low Rank Adaptation (LoRA)

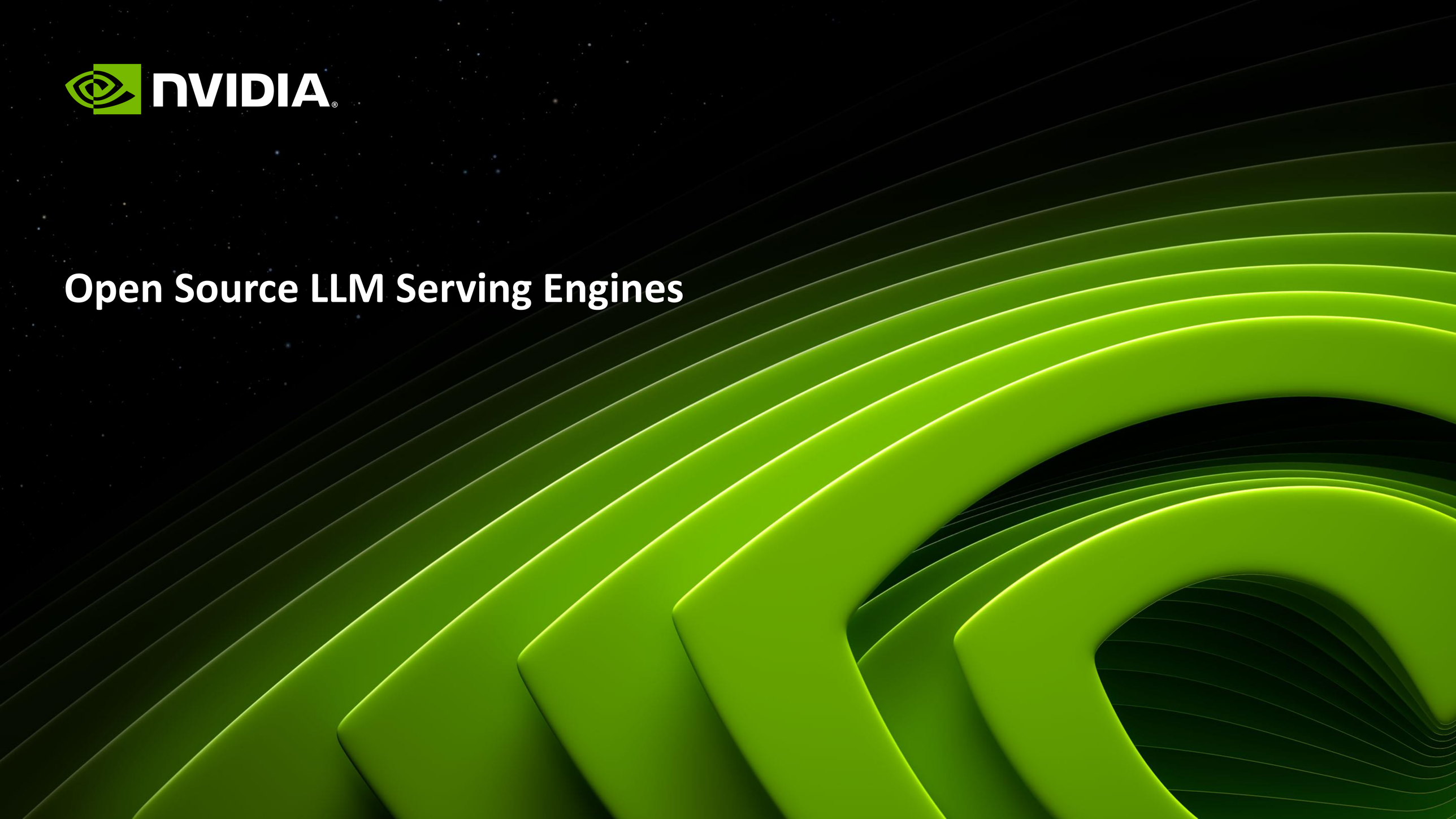
LLM Serving and Deployment


NVIDIA
Custom Endpoints

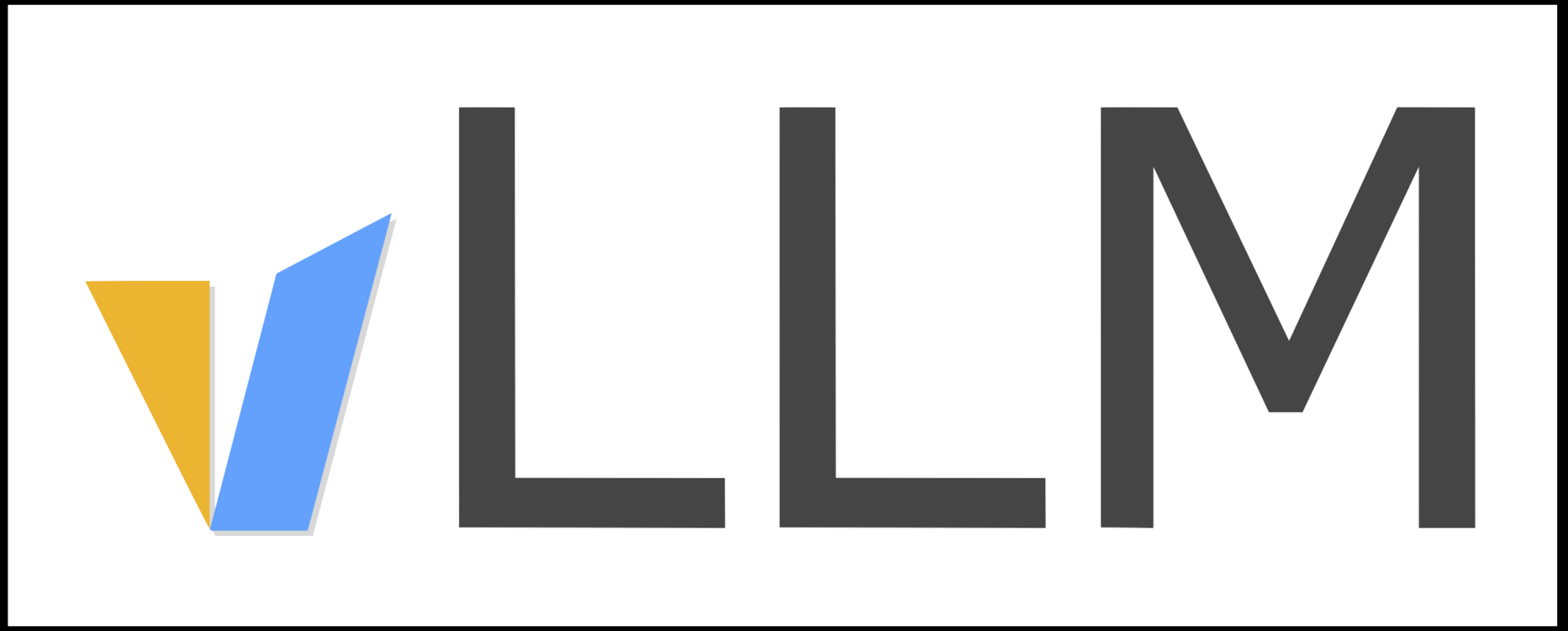
# Custom Endpoints

## Containerized Endpoints

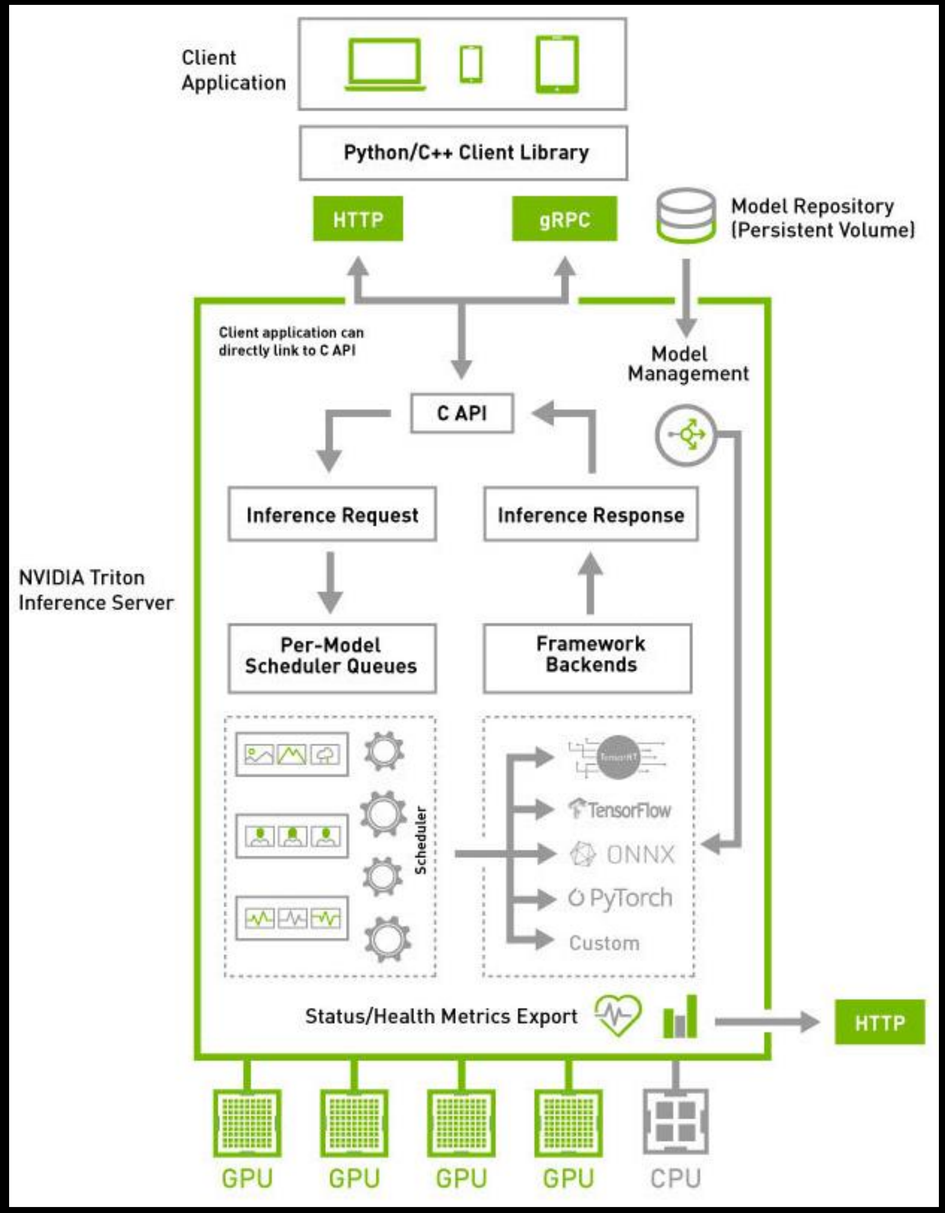



NVIDIA
Open Source LLM Serving Engines

# vLLM

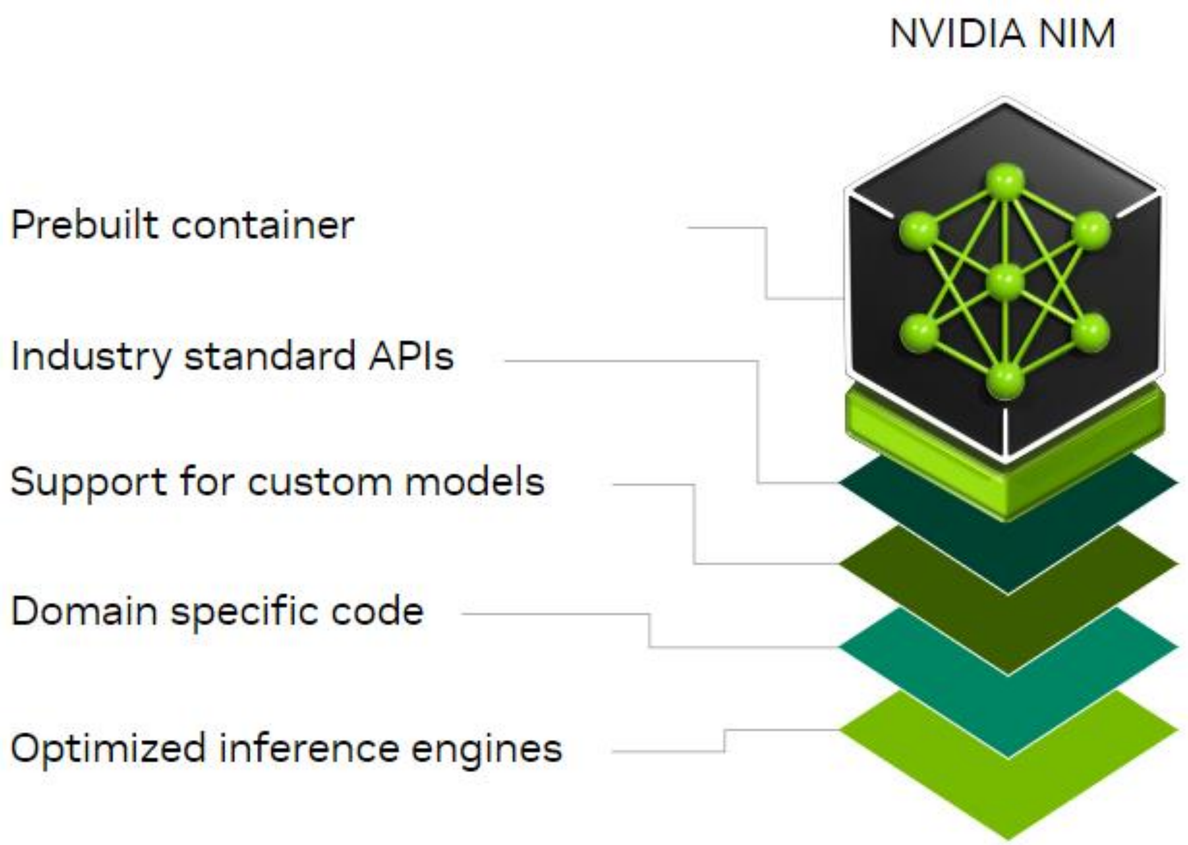


# Nvidia Triton Inference Server



# Nvidia NIMs

Nvidia Inference Microservices




NVIDIA