spring® AI

from exploration to production

Sylvain Puchol

Senior Solution Engineer

VMware Tanzu

Nov 2024

Hello world!

Sylvain Puchol

Solution Engineer at Broadcom | Spring & Tanzu

https://www.linkedin.com/in/sylvain-puchol/

Agenda

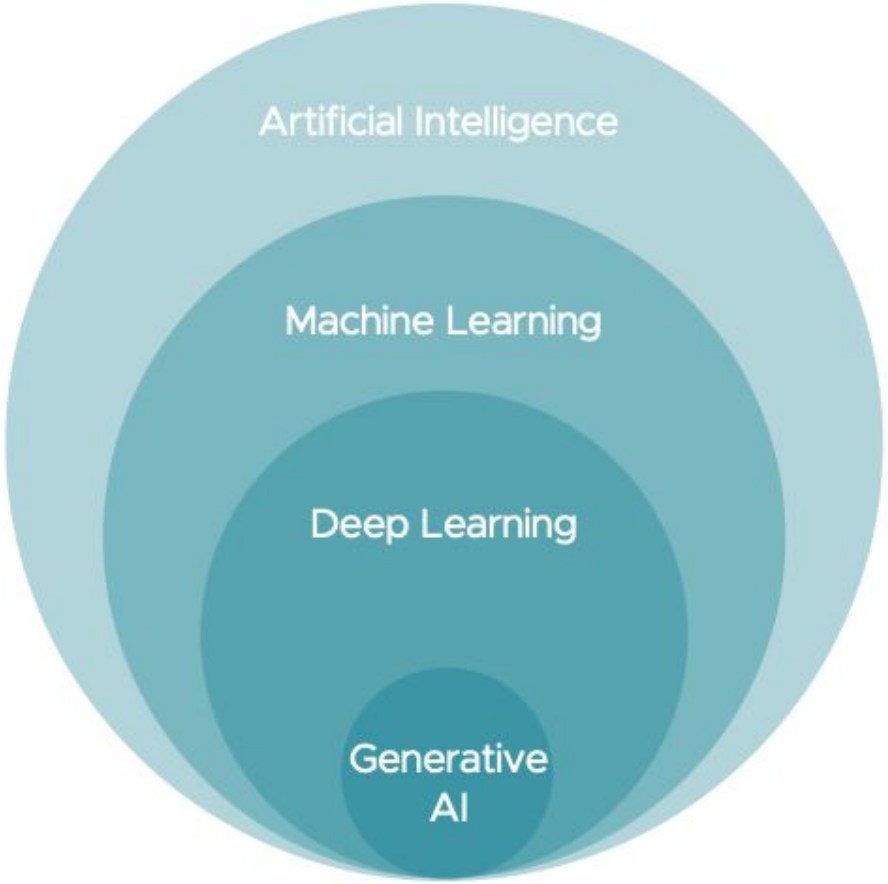
- AI and LLM basics
- Build a LLM-powered App with Spring AI
- Private AI path to production with Tanzu Platform

spring® AI

AI and LLM Basics

GenAI and LLMS

Fundamentals



Artificial Intelligence: Machines are capable of performing cognitive functions typically associated with human minds (chatbot, Boston Dynamics)

Machine learning: Algorithms that learn from data to make predictions or decisions without being explicitly programmed (Classification, Data prediction, Finding patterns, Group similar things)

Deep Learning: Algorithms that simulate how the human brain's neurons work. It adds layers (Neural Networks) between input and output data to learn at much depth (Deepfake)

Generative AI is capable of generating text, images, or other data by utilizing models that learn patterns and structure of their training data

GenAI and LLMS

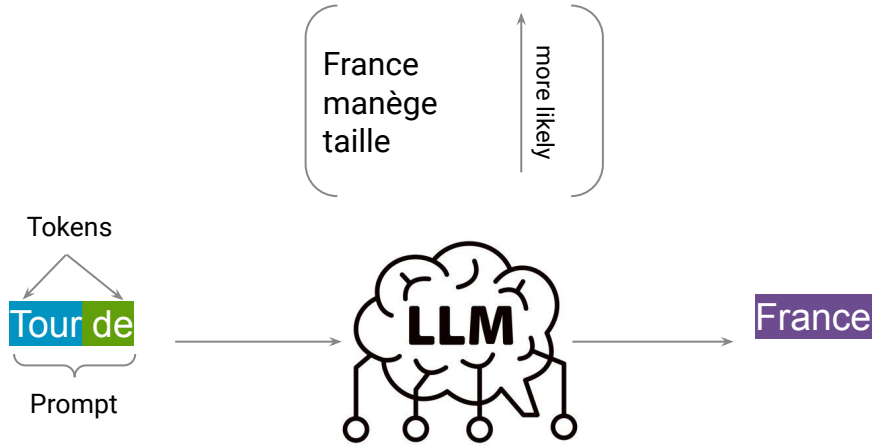
Fundamentals

Machine Learning Model: A mathematical model trained on a specific dataset to make predictions or classifications on new data

Foundation Model: An ML model trained on a huge amount of generic data that serves as the base for various generative tasks

Large Language Models (LLMs): AI models specifically designed to understand and generate human language

- Prompts: Input instructions or data given to the model to guide content generation
- Tokens: Basic units of data processed by models, such as words or parts of words in text generation



Using a foundational/public LLM for your Apps

How could it help with your business problems

Strengths

- Large general knowledge base
- Beats the majority of human level scores in exams
- Can align responses based on your goals
- Ability to create new original content
- Multiple modalities
- Instant summarization

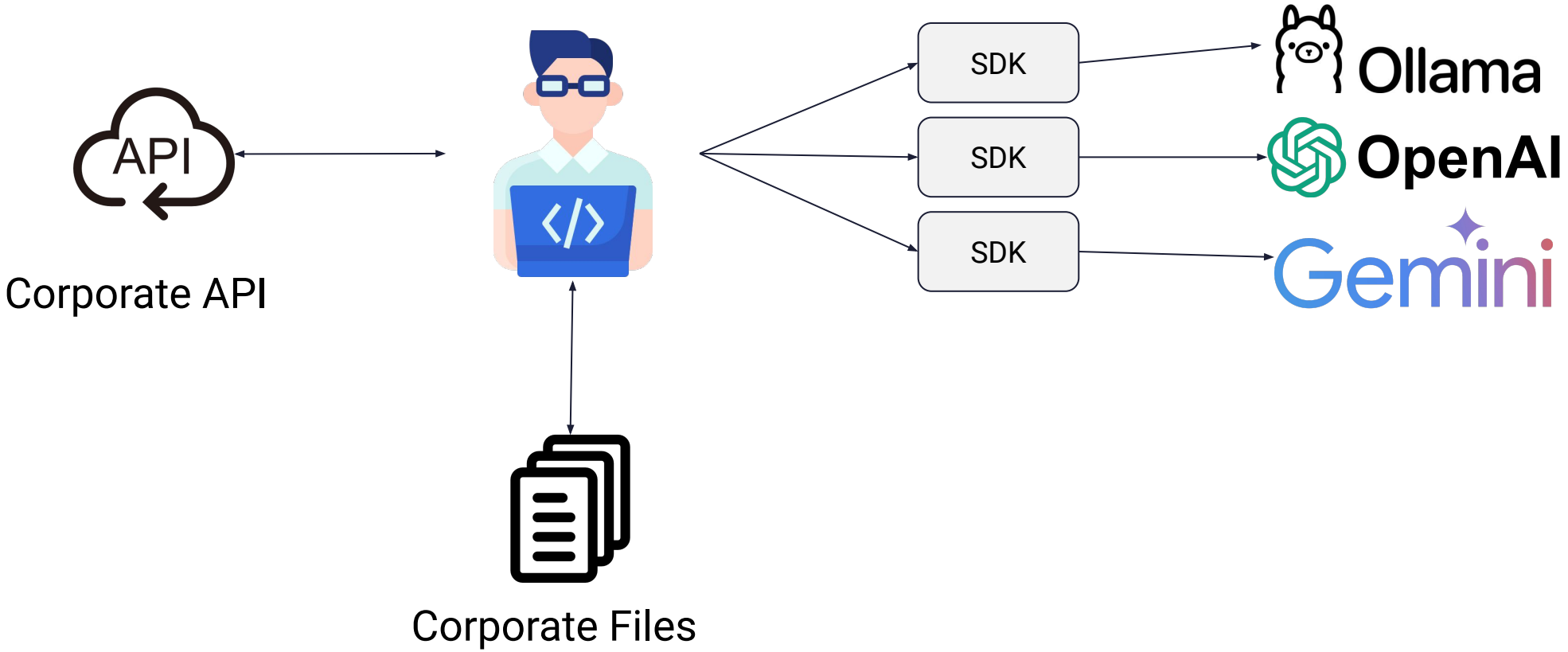
Weaknesses

- Stateless
- Not aware of your API's
- Not trained on your Data
- High training costs/time to fill the knowledge gap
- No structured output

vmware by Broadcom

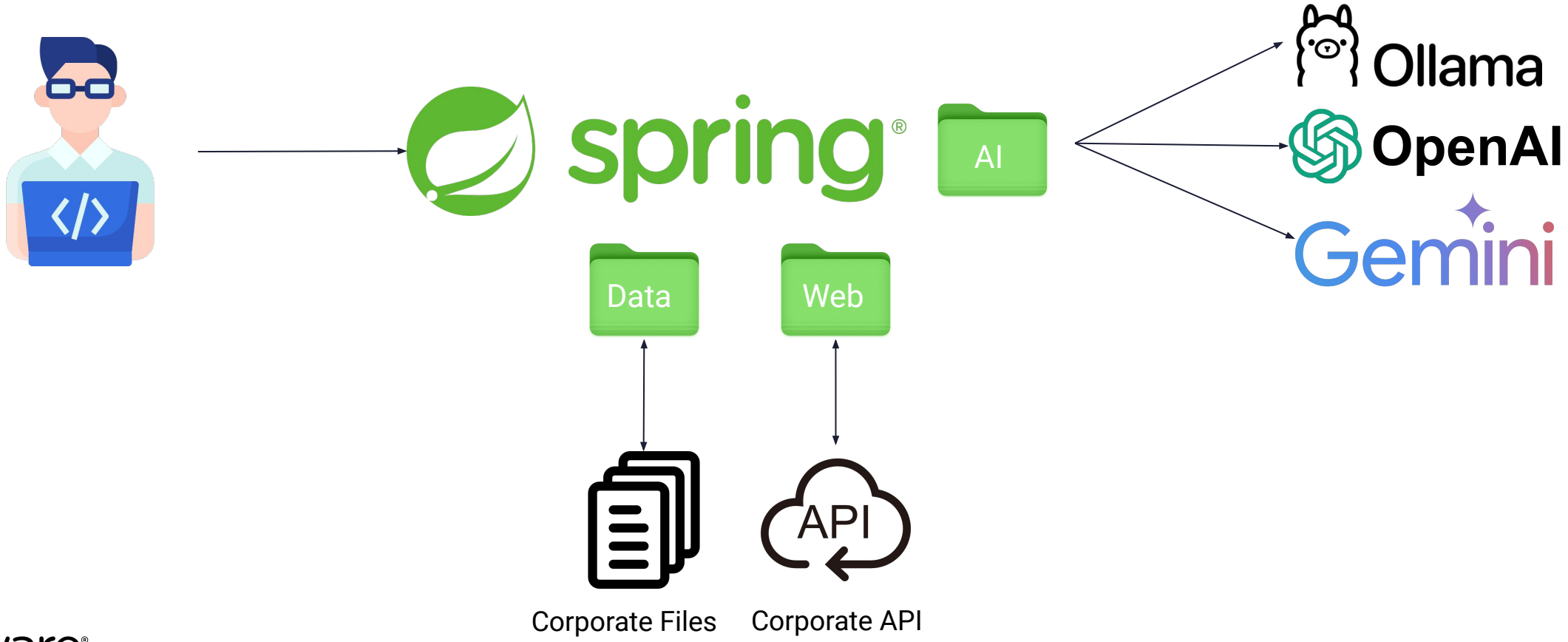
Using a foundational/public LLM on your own

You are the man-in-the-middle between LLM and your Corporate Data

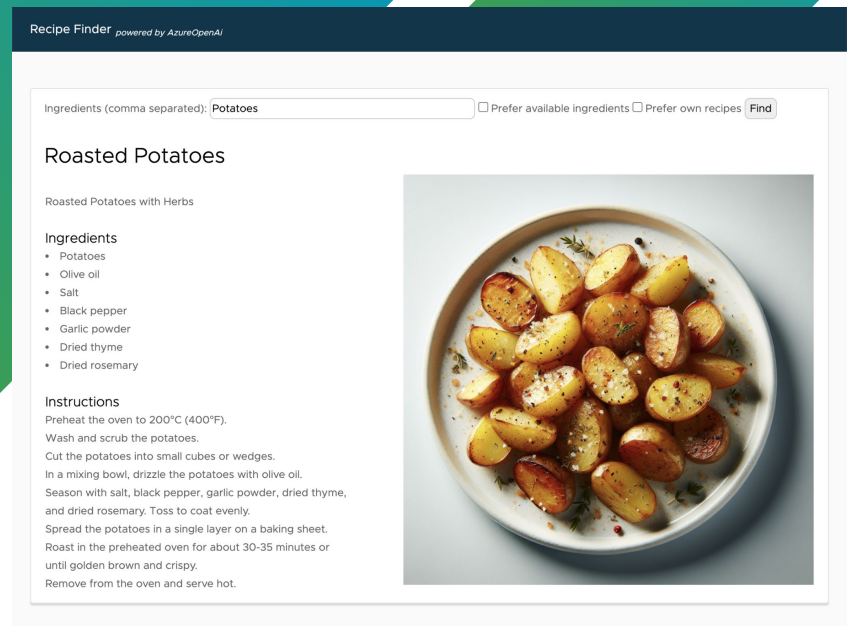


Using Spring AI to cope with LLM challenges

Standardization and simplification with the spring ecosystem



Build a LLM-Powered business application with Spring AI



Challenges and constraints for my business application

spring AI

Provider locking	I want my app to use any LLM
Align responses to goals	I want an engaging chatbot talking like a chef
Not trained on your data	LLM needs to have access to a pdf cookbook
No structured input/output	User inputs has to be used. A recipe java object is defined
Not aware of your Business Logic	Function giving available ingredients in the kitchen has to be used
Work with more than text	I want the users to be able to scan a dish a get its recipe

vmware by Broadcom

Spring AI makes your code portable

No LLM provider locking

Provider	Multimodality	Tools/Functions	Streaming	Retry	Observability	Built-in JSON	Local	OpenAI API Compatibility
Anthropic Claude	text, image	✔	✔	✔	✔	✘	✘	✘
Azure OpenAI	text, image	✔	✔	✔	✔	✔	✘	✔
Google VertexAI Gemini	text, image, audio, video	✔	✔	✔	✔	✔	✘	✔
Google VertexAI PaML2	text	✘	✘	✘	✘	✘	✘	✘
Groq (OpenAI-proxy)	text, image	✔	✔	✔	✔	✘	✘	✔
HuggingFace	text	✘	✘	✘	✘	✘	✘	✘
Mistral AI	text	✔	✔	✔	✔	✔	✘	✔
MiniMax	text	✔	✔	✔	✔	✘	✘	
Moonshot AI	text	✘	✔	✔	✔	✘	✘	
NVIDIA (OpenAI-proxy)	text, image	✔	✔	✔	✔	✘	✘	✔
Ollama	text, image	✔	✔	✔	✔	✔	✔	✔
OpenAI	text, image	✔	✔	✔	✔	✔	✘	✔
QianFan	text	✘	✔	✔	✔	✘	✘	✘
ZhiPu AI	text	✔	✔	✔	✔	✘	✘	✘
Watsonx.AI	text	✘	✔	✘	✘	✘	✘	✘
Amazon Bedrock/Cohere	text	✘	✔	✘	✘	✘	✘	✘
Amazon Bedrock/Jurassic	text	✘	✘	✘	✘	✘	✘	✘
Amazon Bedrock/Llama	text	✘	✔	✘	✘	✘	✘	✘
Amazon Bedrock/Titan	text	✘	✔	✘	✘	✘	✘	✘
Amazon Bedrock/Anthropic 3	text	✘	✔	✘	✘	✘	✘	✘

Source: https://docs.spring.io/spring-ai/reference/api/chat/comparison.html

Let’s solve these LLM challenges with 

- Provider locking → SpringAI API architecture
- Align responses to goals
- Not trained on your data
- No structured input/output
- Not aware of your Business Logic
- Work with more than text

Easy chatbot through Prompt API

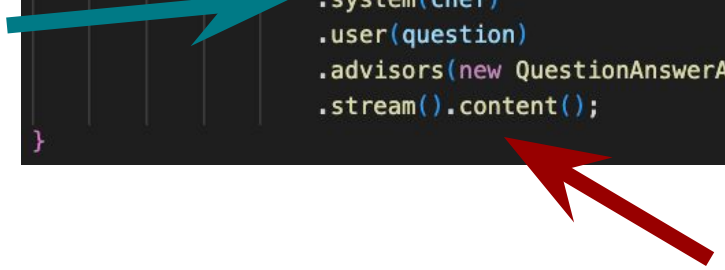
One liner - you don’t need to know about the implementation

```
@GetMapping(value = "/hello", produces = MediaType.TEXT_PLAIN_VALUE)
String chat(@RequestParam("q") String query) {
    // A single line API call to connect to your favorite LLM and get a response.
    return chatClient.prompt().user(query).call().content();
}
```

Let’s align the model to your goals

System prompt and content streaming

```
public Flux<String> answerCustomerQuestion(String question) {
    log.info(msg:"Answering question from Chat");
    var advisorSearchRequest = SearchRequest.query(question).withTopK(topK:1).withSimilarityThreshold(threshold:0.8);
    var advise = new PromptTemplate(chefCookBook).getTemplate();
    return chatClient.prompt()
                .system(chef)
                .user(question)
                .advisors(new QuestionAnswerAdvisor(vectorStore, advisorSearchRequest, advise))
                .stream().content();
}
```



- **System prompt** gives to the LLM guidances on his persona and behavior
- **Stream** parameter allows us to get tokens on the fly. User feels like he is chatting with somebody that is typing the answer

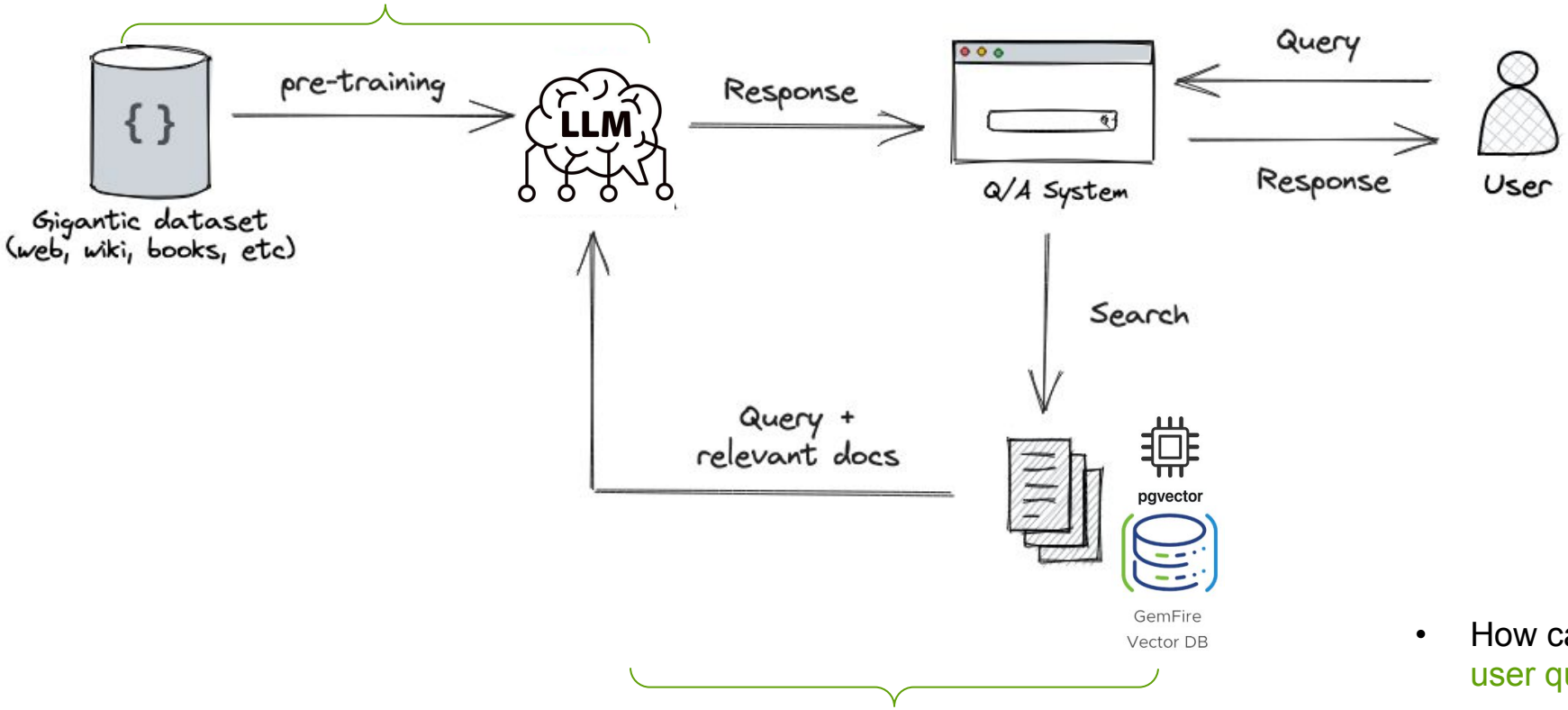
Let’s solve these LLM challenges with Spring AI

- Provider locking → SpringAI API architecture
- Align responses to goals → System prompt, streaming
- Not trained on your data
- No structured input/output
- Not aware of your Business Logic
- Work with more than text

Give LLM access to your Data

Retrieval augmented Generation: LLM knowledge without additional training

1. Let the LLM provider train a model with generic skills



2. Provide a meaningful context to this model when you call it

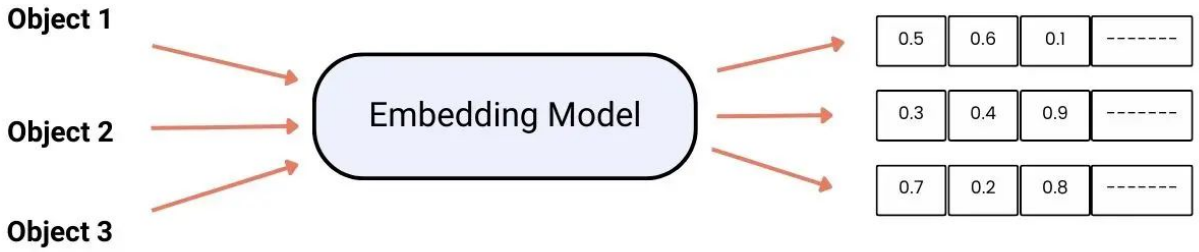
- How can I ensure to give data related to the user query?
- How can I avoid to slow down the overall process if my dataset is gigantic and unevenly structured?

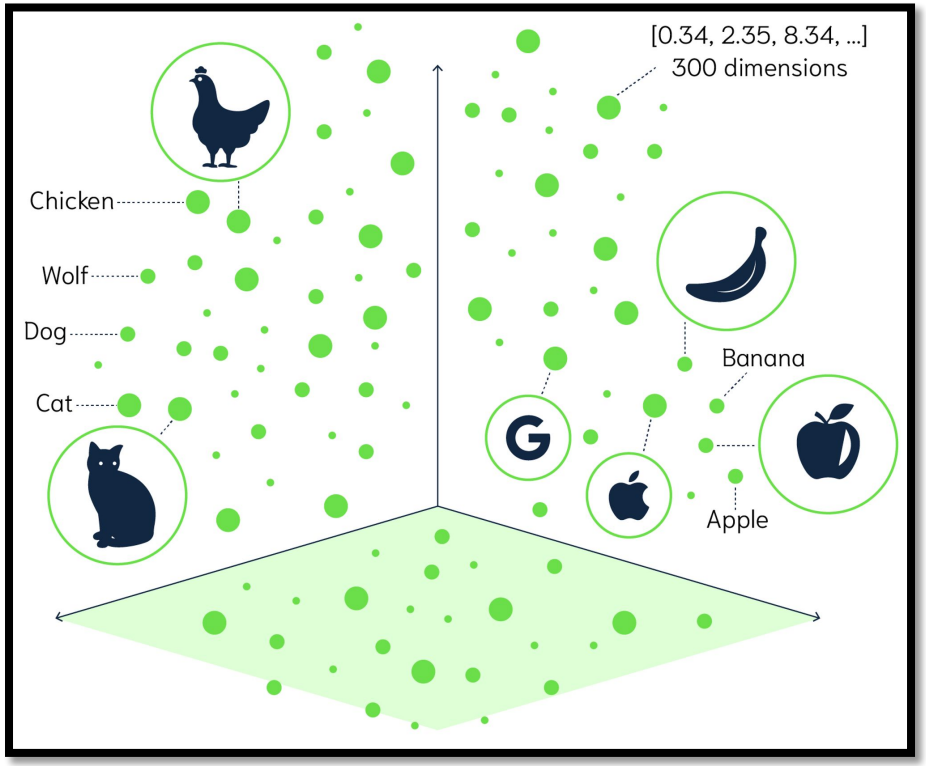
Give LLM access to your Data

A zoom on vector Databases and quantization

Vector Embeddings enable an LLM to understand the meaning and context of data.

- Quantization process changes data into Vector (list) of floating-point numbers
- Distance between two vectors measures their relatedness (SimilaritySearch)

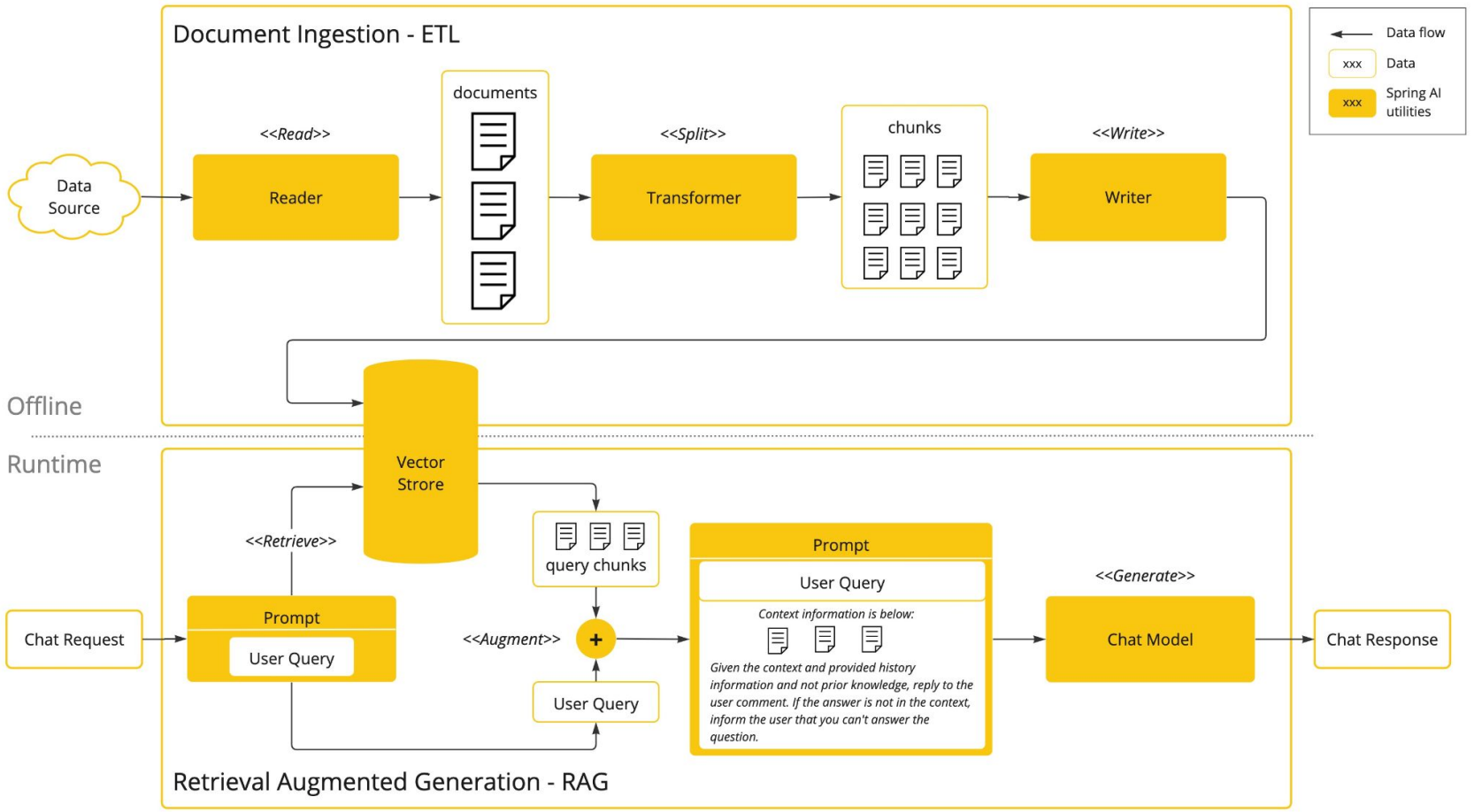




Source: weaviate.io

Spring AI makes RAG easy

SpringAI manages your ETL and vector stores



Spring AI provides:

- A Vector Store API, enabling portability across different vector database providers
- ETL(Extract Transform and Load) framework to populate the vector database with embeddings
- A Chat/Embeddings Model API with support for several models

Spring AI makes RAG easy

Give LLM data to perform RAG, along with behavior guidance

```
public Flux<String> answerCustomerQuestion(String question) {
    log.info(msg:"Answering question from Chat");
    var advisorSearchRequest = SearchRequest.query(question).withTopK(topK:1).withSimilarityThreshold(threshold:0.8);
    var advise = new PromptTemplate(chefCookBook).getTemplate();
    return chatClient.prompt()
                     .system(chef)
                     .user(question)
                     .advisors(new QuestionAnswerAdvisor(vectorStore, advisorSearchRequest, advise))
                     .stream().content();
}
```

- **advisorSearchRequest** controls parameters of similar data retrieval on vectorDB
- **Advise** parameter provides guidance on what to do with retrieved data

Let’s solve these LLM challenges with Spring AI

Provider locking	SpringAI API architecture
Align responses to goals	System Prompt
Not trained on your data	Spring ETL and Advisors for RAG
No structured input/output	
Not aware of your Business Logic	
Work with more than text	

Prompt templating

Build dynamic prompts based on your users input

Prompt template

```
Provide a recipe that includes in the best case all of the following
ingredients.

Ingredients: """
{ingredients}
"""

Add additional ingredients that are necessary for a good flavor or to create a
more creative and complex meal.

The recipe should be translated to English, and with quantity in metric system.
```

Prompt rendered from parameters

```
@Value("classpath:/prompts/recipe-for-ingredients")
private Resource recipeForIngredientsPromptResource;



return chatClient.prompt()
        .user(us -> us
                .text(recipeForIngredientsPromptResource)
                .param("ingredients", String.join(",", ingredients)))
        .call()
        .content();
```

Response formatting

Ask your LLM to generate a structured response - and let Spring AI maps it to a Java construct

```java
public record Recipe(String name, String description, List<String> ingredients, List<String> instructions, String imageUrl) {

    public Recipe(Recipe recipe, String imageUrl) {
        this(recipe.name, recipe.description, recipe.ingredients, recipe.instructions, imageUrl);
    }
}
```

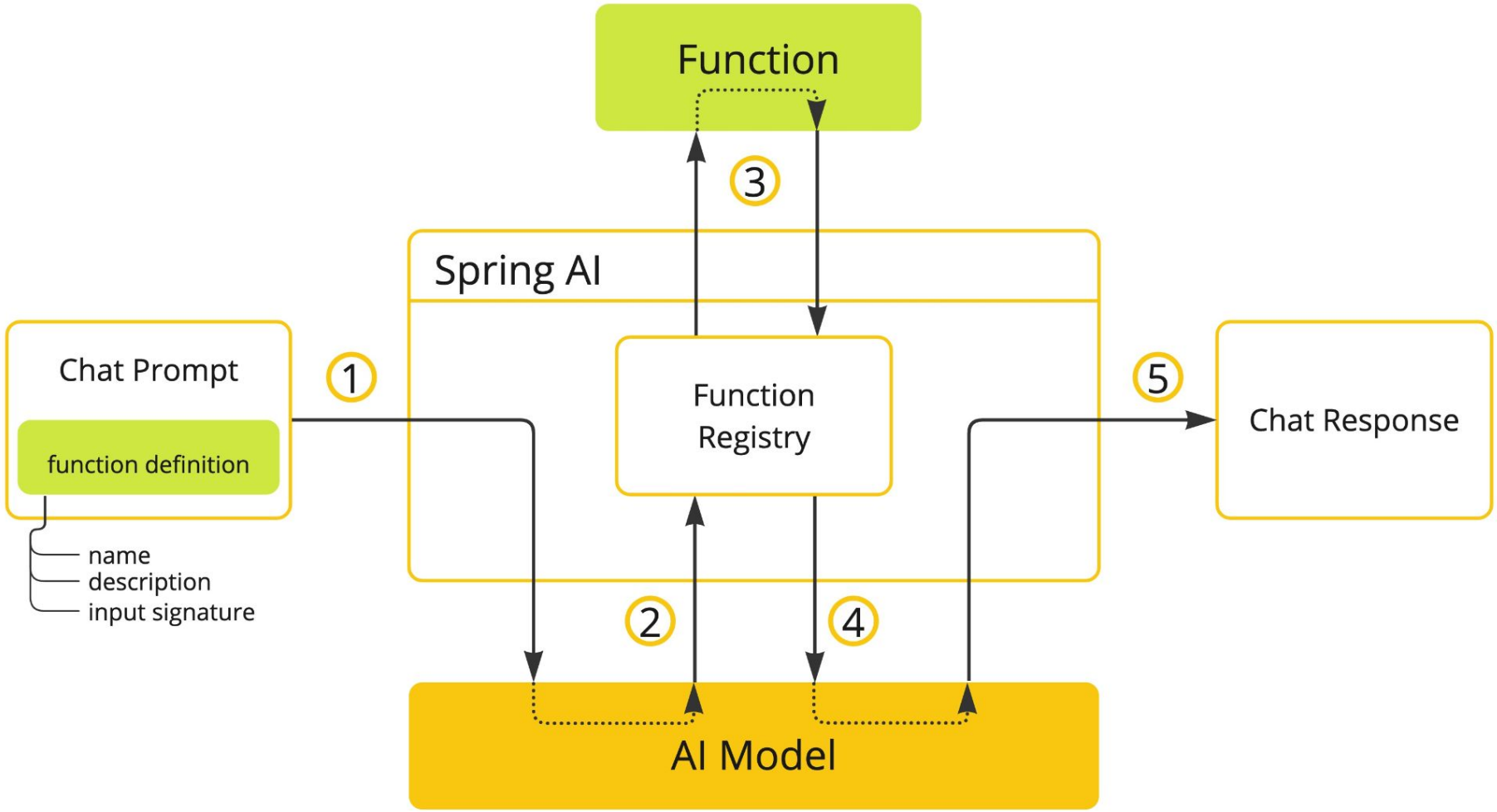
```java
private Recipe fetchRecipeFor(List<String> ingredients) {
    log.info(msg:"Fetch recipe without additional information");

    return chatClient.prompt()
            .user(us -> us
                    .text(recipeForIngredientsPromptResource)
                    .param(k:"ingredients", String.join(delimiter:",", ingredients)))
            .call()
            .entity(type:Recipe.class);
}
```

Let’s solve these LLM challenges with 

Provider locking	SpringAI API architecture
Align responses to goals	System Prompt
Not trained on your data	Spring ETL and advisors for RAG
No structured input/output	Prompt templating / Output Converters
Not aware of your Business Logic	
Work with more than text	

Function API

Let the LLM call your Java methods as needed



Function API in action

Let the LLM call your Java methods as needed

Describe function for LLM guidance

```
@Description("Fetches ingredients that are available at home")
@Service("fetchIngredientsAvailableAtHome")
public class FetchIngredientsAvailableAtHomeFunction implements Function<FetchIngredientsAvailableAtHomeFunction.Re

    private static final Logger log = LoggerFactory.getLogger(clazz:FetchIngredientsAvailableAtHomeFunction.class);

    private final List<String> alwaysAvailableIngredients;
    private final List<String> availableIngredientsInFridge;

    public FetchIngredientsAvailableAtHomeFunction(@Value("${app.always-available-ingredients}") List<String> alway
                                                   @Value("${app.available-ingredients-in-fridge}") List<String> av
        this.alwaysAvailableIngredients = alwaysAvailableIngredients;
        this.availableIngredientsInFridge = availableIngredientsInFridge;
    }
```

Use function bean in the call

```
Provide a recipe that includes in the best case all of the following ingredients plus
the ingredients available at home which are ordered by highest priority.

Ingredients: """
{ingredients}
"""

Add additional ingredients that are necessary for a good flavor or to create a more
creative and complex meal.

The recipe should be translated to English, and with quantity in metric system.
```

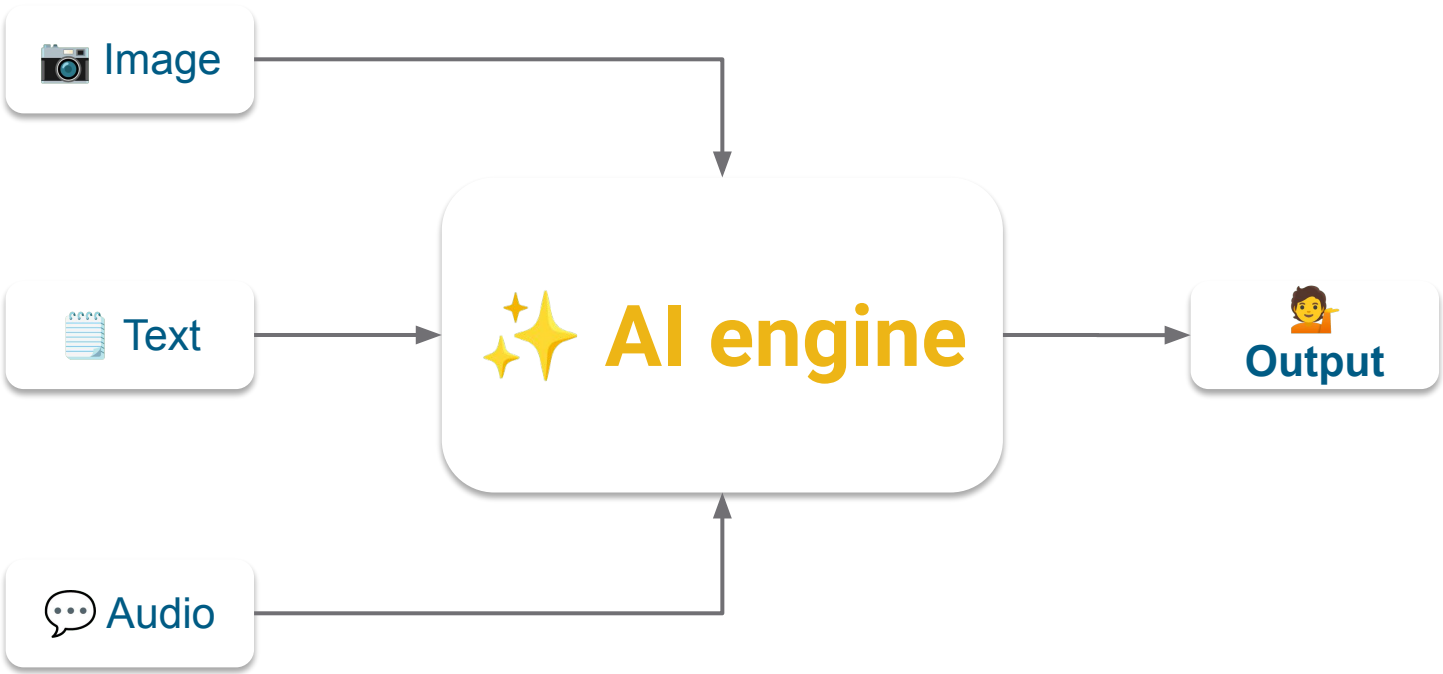
```
return chatClient.prompt()
        .user(us -> us
                .text(recipeForAvailableIngredientsPromptResource)
                .param(k:"ingredients", String.join(delimiter:",", ingredients)))
        .functions(...functionBeanNames:"fetchIngredientsAvailableAtHome")
        .call()
        .entity(type:Recipe.class);
```

Let's solve these LLM challenges with Spring AI

Provider locking	SpringAI API architecture
Align responses to goals	System Prompt
Not trained on your data	Spring ETL and advisors for RAG
No structured input/output	Prompt templating / Output Converters
Not aware of your Business Logic	Function Calling
Work with more than text	

Prompt is not only text

Process and generate text in conjunction with other modalities such as images, audio, or video



Build a multimodal prompt with Spring AI

Include an image to provide context to your text based prompt

```
return chatClient.prompt()
        .user(us -> {
            try {
                us
                    .text("""
                        The following image is a dish,
                        you have to find out what is this dish and then provide the recipe for this dish
                        """)
                    .media(MimeTypeUtils.IMAGE_JPEG, new URI(ingredients.getFirst()).toURL());
```

Let’s solve these LLM challenges with Spring AI

Provider locking	SpringAI API architecture
Align responses to goals	System Prompt
Not trained on your data	Spring ETL and advisors for RAG
No structured input/output	Prompt templating / Output Converters
Not aware of your Business Logic	Function Calling
Work with more than text	Multimodal prompt

Private AI path to production for my application

What do you need to run LLM powered app in production?

Requirements checklist

Container

Ship your application anywhere

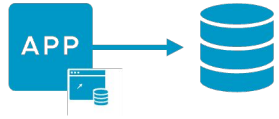
CI/CD

Being to deploy and update your app in minutes

Fault Tolerance

Replicas an http routing

RAG DB



Provide Context to your LLM

Tanzu Platform: SpringAI App lifecycle made easy

A CF push like experience on top of a GPUready cloud

Build

Containerize apps, Scalable on VMs & K8s Runtimes

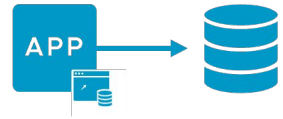
Deploy

Apps instrumented with Platform Capabilities

Scale

API Based Automation for Apps Operations

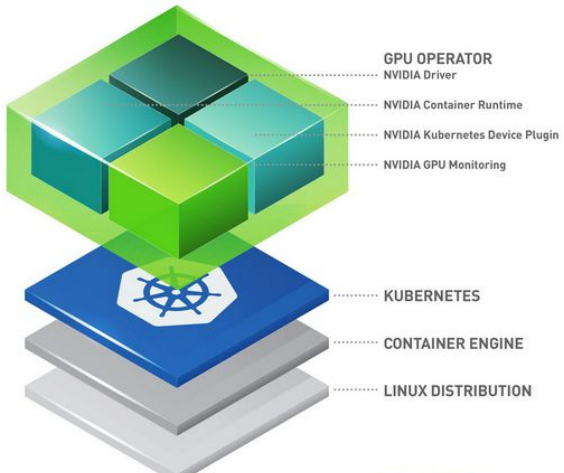
Bind



Bindings to DBaaS, OSS/Cloud Catalogs

vLLM

Hugging Face



vmware by Broadcom

Private AI path to production Demo: details

```
tanzu deploy
```

```
$ ls
README.md package.json server.js my-app/
$ pack build paketo-demo-app
```

paketobuildpacks/
builder:base
with Ubuntu 18.04 OS

< your app >

VMware Tanzu® Platform

Openai compatible API

BIND RAGdb

bitnami
Data services

recipe-finder

Tanzu managed Namespace

vLLM

MISTRAL AI_

Hugging Face

LLM namespace

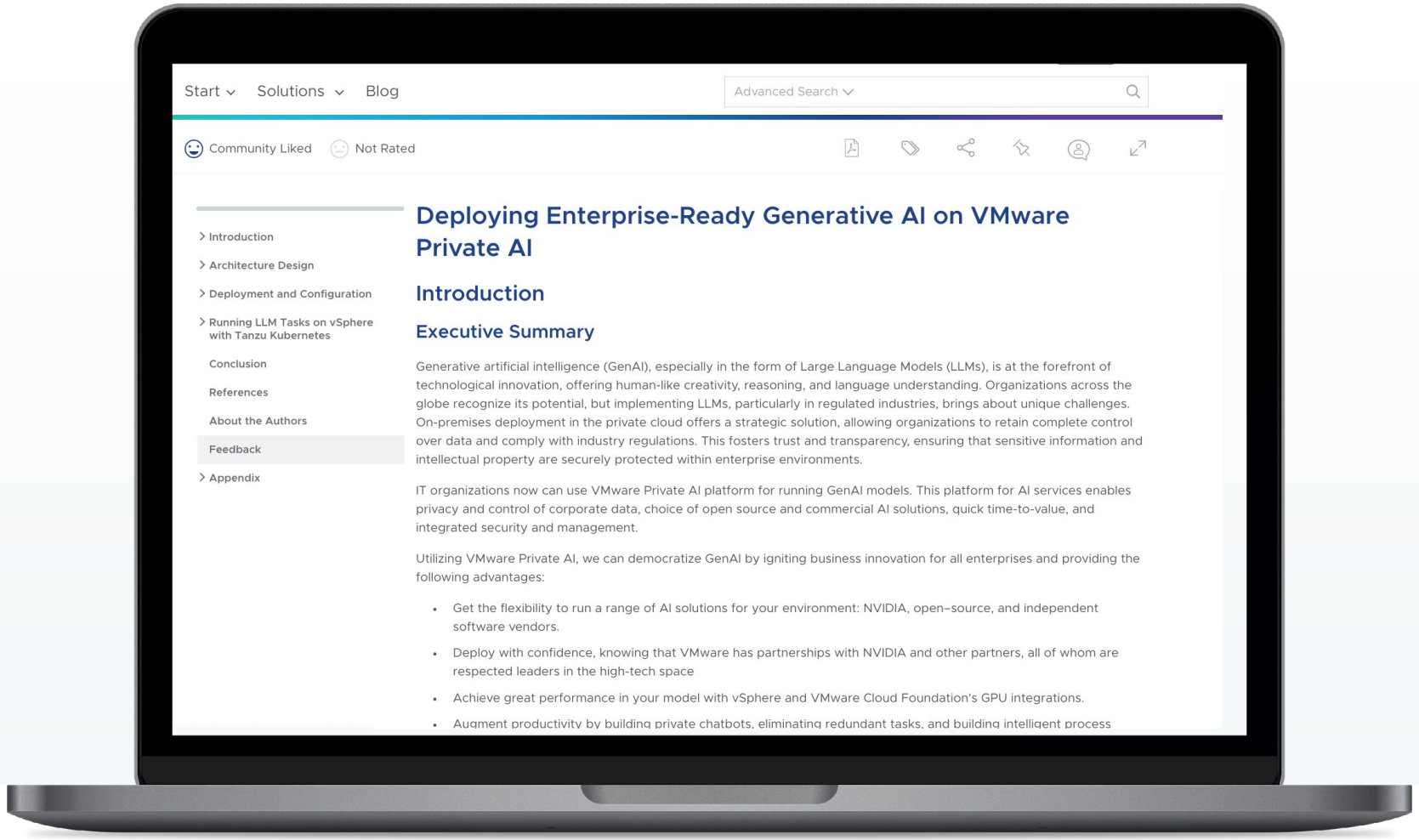
GPU

Production EKS Cluster

34

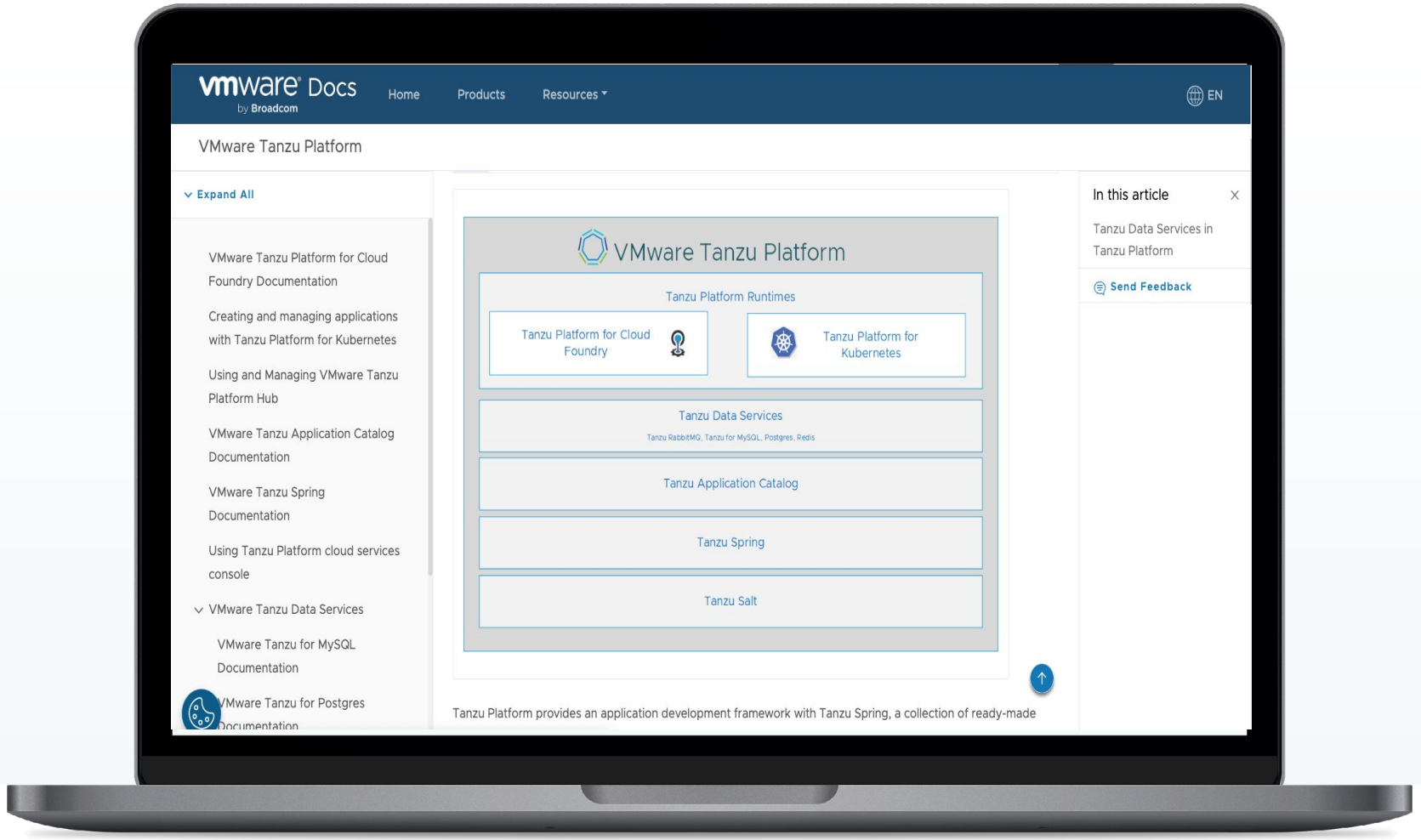
VMware Private AI reference architecture

Deepdive on training and inference on any k8s

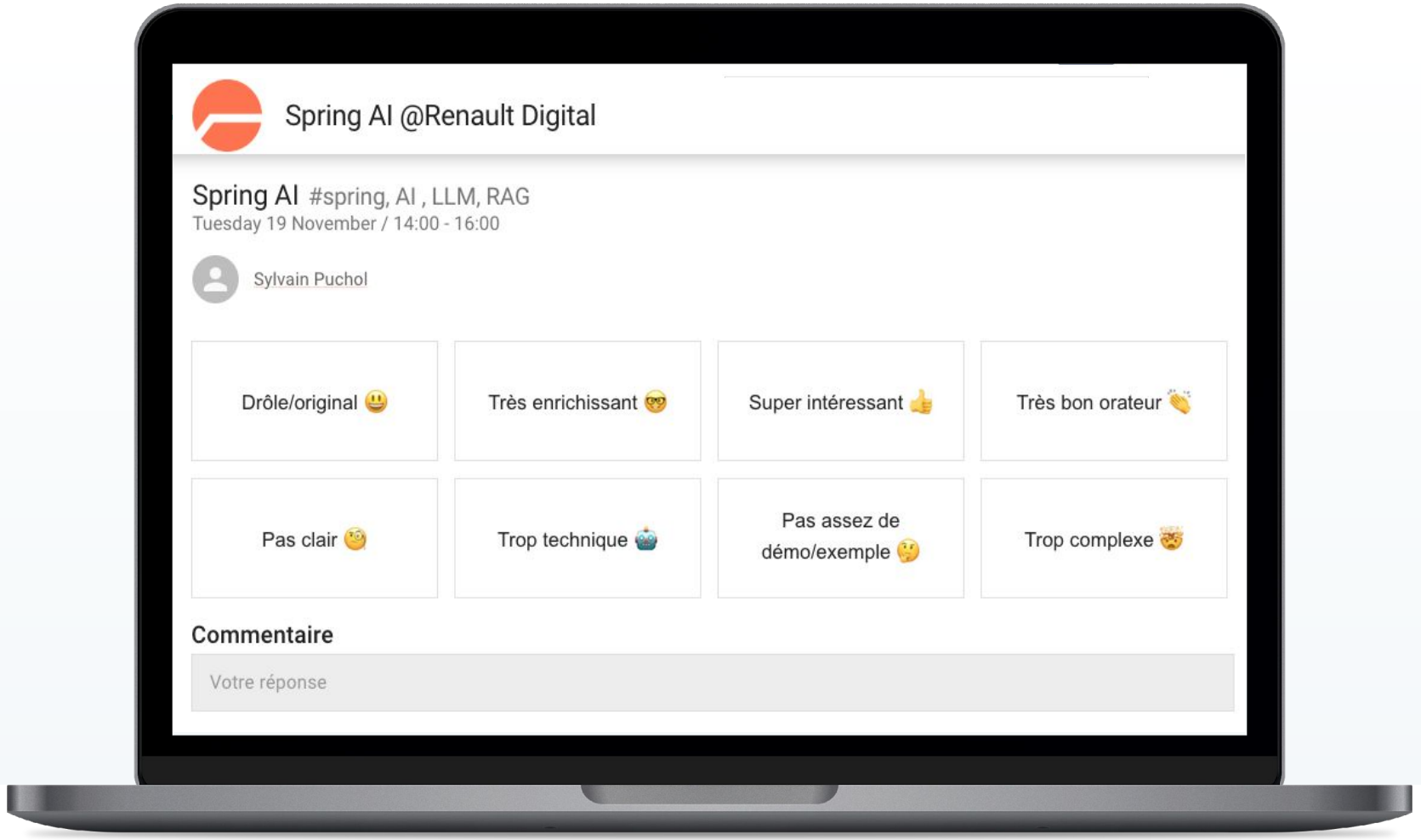


Tanzu platform overview

Build bind an scale your AI apps to any k8s



#opentofeedback



Thank You

Let’s keep in touch:

Sylvain Puchol

https://www.linkedin.com/in/sylvain-puchol/

vmware by Broadcom