



# **GenAI with RAG architecture and beyond on GCP**

Marek Wiewiórka, PhD  
Krzysztof Zarzycki

# About us



**Marek Wiewiórka**

Chief Architect @ Xebia Data PL



**Krzysztof Zarzycki**

CTO @ Xebia Data PL

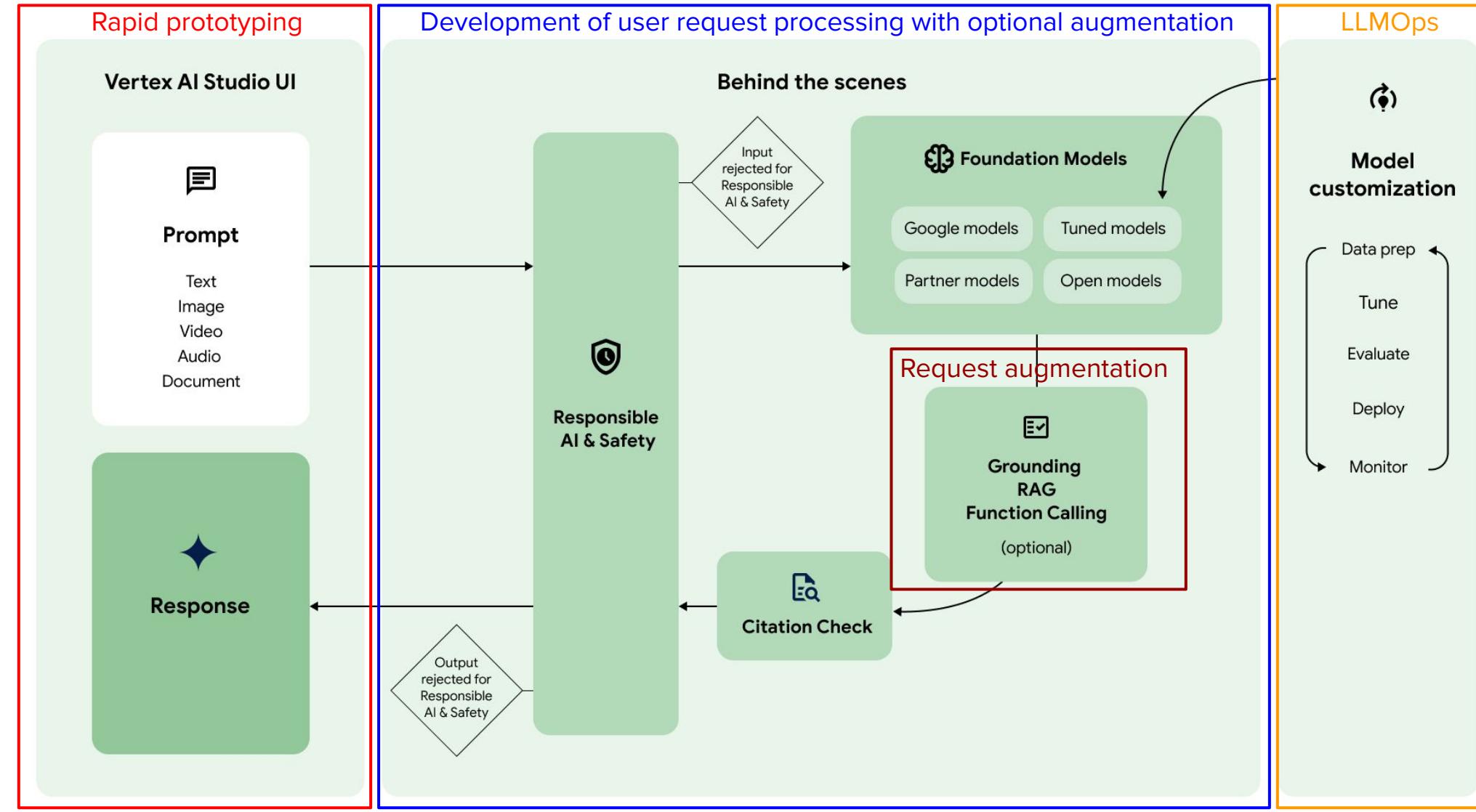


# Agenda

1. Quick start with Generative AI in GCP
2. Prompt and context augmentation techniques
3. RAG flavours jump starts for GCP
4. Going live!

Encore: DEMO of Xebia Data Copilot for Analytics Engineers

# Generative AI workflow in Vertex AI





# Quick start

# Prototyping with Vertex AI Studio - no code approach

The screenshot shows the Vertex AI Studio interface for prototyping. The left sidebar lists categories like Colab Enterprise, Workbench, Overview, Multimodal (selected), Language, Vision, Speech, Feature Store, Datasets, Labeling tasks, Training, Experiments, Metadata, and Marketplace. The main area has tabs for System instructions, Prompt, and Response. The Prompt tab contains a question: "What is the first opponent of Hurkacz in Roland Garros 2024 ?". Below it is a "Select source" dropdown with options: "Upload from computer" (Small files 7MB and under) and "Import from Cloud Storage" (For large files over 7MB, longer process times apply). A red box highlights this section. To the right is a "CLEAR PROMPT" button and an "INSERT MEDIA" button with a plus sign. The Response tab shows the generated answer: "Hubert Hurkacz's first opponent in Roland Garros 2024 is Shintaro Mochizuki. [1]" and a "Grounding Sources" link. A purple arrow points from the "Grounding Sources" link to a purple box at the bottom right. The top navigation bar includes Google Cloud, Generative Language Client, a search bar, and user profile icons. The right side of the interface shows model configuration: Model (gemini-1.5-flash-001), Region (us-central1 (Iowa)), Temperature (1), Output token limit (8192), Safety settings (with Hate speech, Dangerous content, Sexually explicit content, and Harassment content dropdowns), Advanced settings (Max responses: 8, Top-K: 40, Top-P: 1), and a "Enable Grounding" checkbox (SOURCE: Google search, CUSTOMIZE). A blue arrow points from the "SAFETY SETTINGS" button to the "Harmful content" dropdown. A red box highlights the "Model" and "Region" configuration.

System instructions

Select source

- Upload from computer Small files 7MB and under
- Import from Cloud Storage For large files over 7MB, longer process times apply

CLEAR PROMPT

INSERT MEDIA

Safety settings

You can adjust the likelihood of receiving a model response that could contain harmful content. Content is blocked based on the probability that it's harmful. [Learn more](#)

Hate speech  
Block some

Dangerous content  
Block some

Sexually explicit content  
Block some

Harassment content  
Block some

RESET DEFAULTS

SAVE

CLOSE

Model: gemini-1.5-flash-001

Region: us-central1 (Iowa)

Temperature: 1

Output token limit: 8192

Add stop sequence

Press Enter after each sequence

SAFETY SETTINGS

Advanced

Max responses: 8

Top-K: 40

Top-P: 1

Enable Grounding

SOURCE: Google search

CUSTOMIZE

# Prototyping with Notebooks (Colab/Workbench) - low code approach



[Run in Colab](#) [View on GitHub](#) [Open in Vertex AI Workbench](#)

```
grounding_source = GroundingSource.WebSearch()

response = text_model.predict(
    PROMPT,
    grounding_source=grounding_source,
)

response, response.grounding_metadata
```

## Initialize Vertex AI SDK for Python

Initialize the Vertex AI SDK for Python for your project:

```
In [8]: vertexai.init(project=PROJECT_ID, location=REGION)
```

Initialize the generative text and chat models from Vertex AI:

```
In [9]: text_model = TextGenerationModel.from_pretrained("text-bison")
chat_model = ChatModel.from_pretrained("chat-bison")
```

## Example: Grounding with Google Search results

In this example, you'll compare LLM responses with no grounding with responses that are grounded in the results of a Google Search. You'll ask a question about a recent hardware release from the Google Store.

```
In [10]: PROMPT = (
    "What are the price, available colors, and storage size options of a Pixel Tablet?"
)
```

### Text generation without grounding

Make a prediction request to the LLM with no grounding:

```
In [11]: response = text_model.predict(PROMPT)
response
```

```
Out[11]: **Price:**
```

```
* Starting at $399 for the Wi-Fi-only model with 128GB of storage
* $499 for the Wi-Fi + 5G model with 128GB of storage
* $599 for the Wi-Fi + 5G model with 256GB of storage
```

# Gemini model family

May 24, 2024

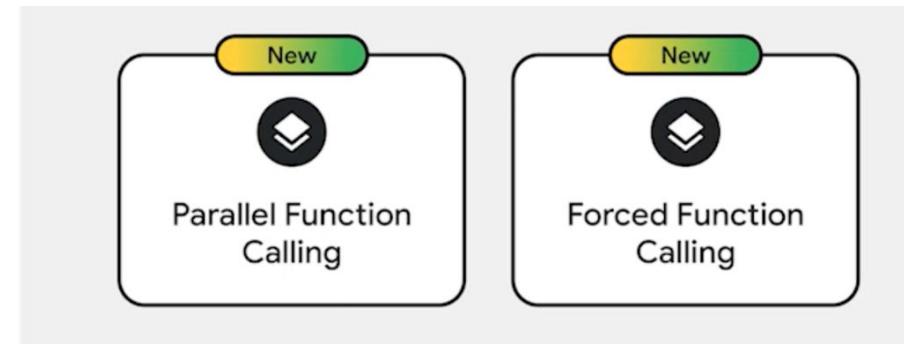
## Gemini 1.5 Pro

Mid-size multimodal model with breakthrough long-context understanding

- A new **Mixture-of-Experts (MoE) architecture** that provides more efficient training and serving, while increasing model performance
- An **expanded context window** (up to 1 million tokens) for complex reasoning across vast amounts of information
- **Better understanding and reasoning across modalities** including text, code, image, audio and video

### ANNOUNCEMENT

The Gemini 1.5 Pro (`gemini-1.5-pro-001`) and Gemini 1.5 Flash (`gemini-1.5-flash-001`) models are **Generally Available**. For more information, see [Google models](#), [Overview of the Gemini API](#), and [Send multimodal prompt requests](#).



Model	Text	Code	Images	Audio	Video	Video/audio	PDF
Gemini 1.5 Flash	✓	✓	✓	✓	✓	✓	✓
Gemini 1.5 Pro	✓	✓	✓	✓	✓	✓	✓
Gemini 1.0 Pro Vision	✓	✓	✓	✗	✓	✗	✓
Gemini 1.0 Pro	✓	✓	✗	✗	✗	✗	✗

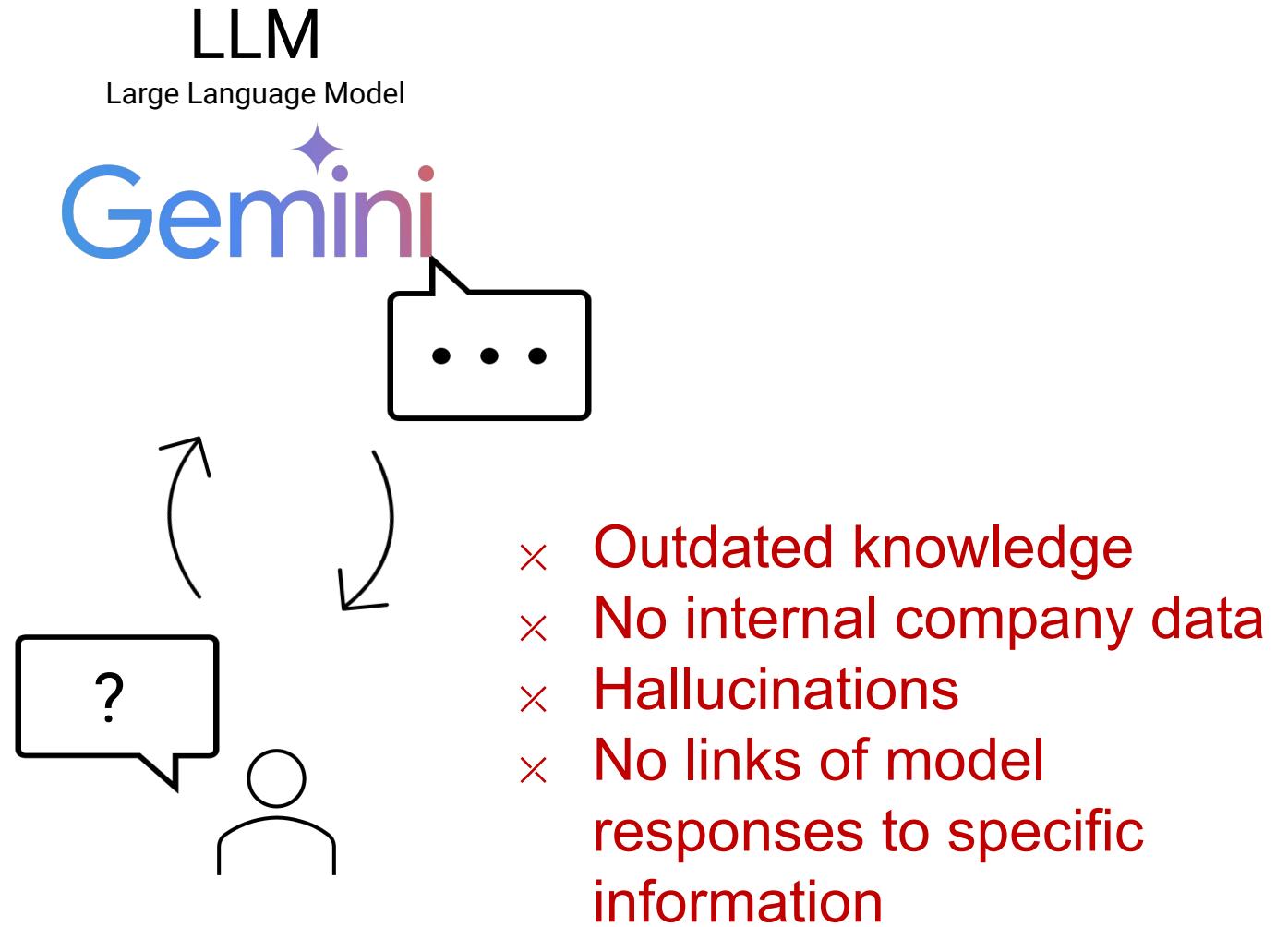


# Prompt and context augmentation techniques

# Key LLM-powered apps challenges

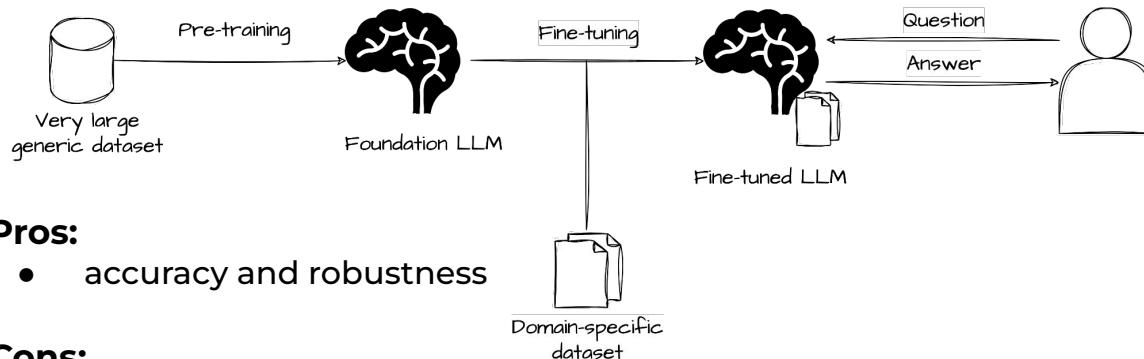
LLM	Knowledge cut-off
Gemini 1.5 Pro/Flash	Early 2023 <a href="#">[1]</a>
Gemini 1.0 Pro	Early 2023 <a href="#">[1]</a>
GPT-3.5 turbo	September 2021 <a href="#">[2]</a>
GPT-4	September 2021 <a href="#">[3]</a>

Knowledge cut-offs as of May 2024.



# How to overcome some of them?

## Full fine-tuning



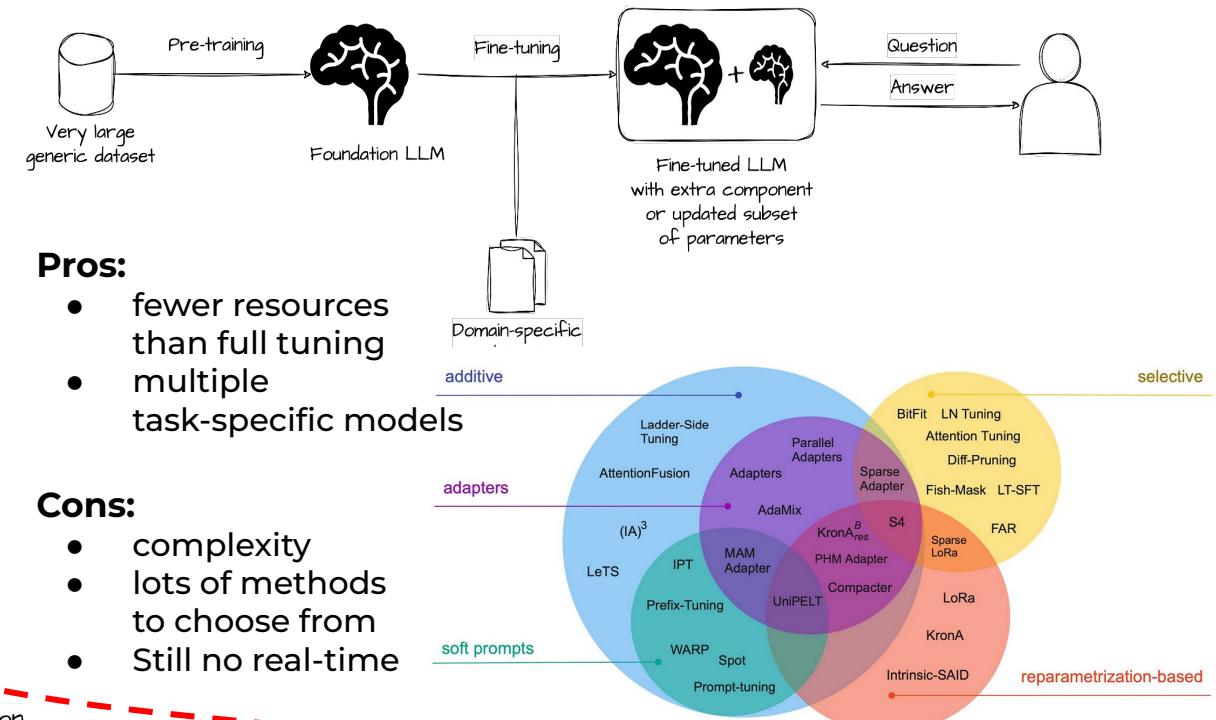
### Pros:

- accuracy and robustness

### Cons:

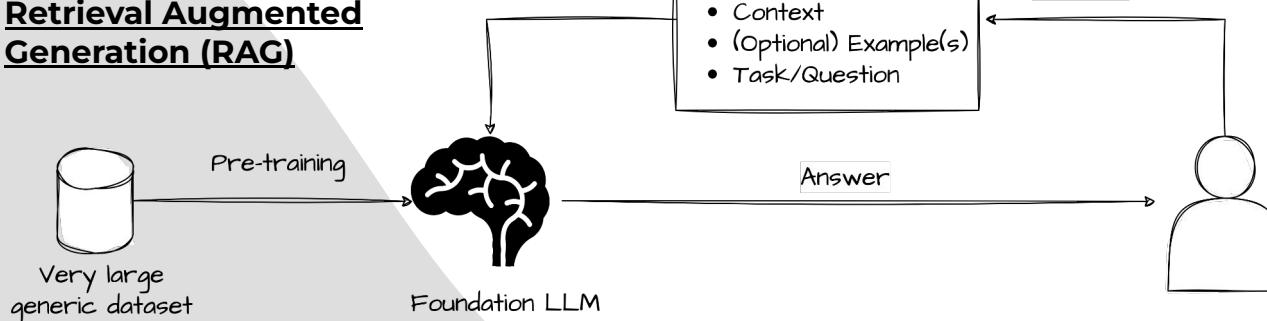
- cost (time)
- hardware requirements (memory)
- time

## Parameter-efficient fine tuning (PEFT)

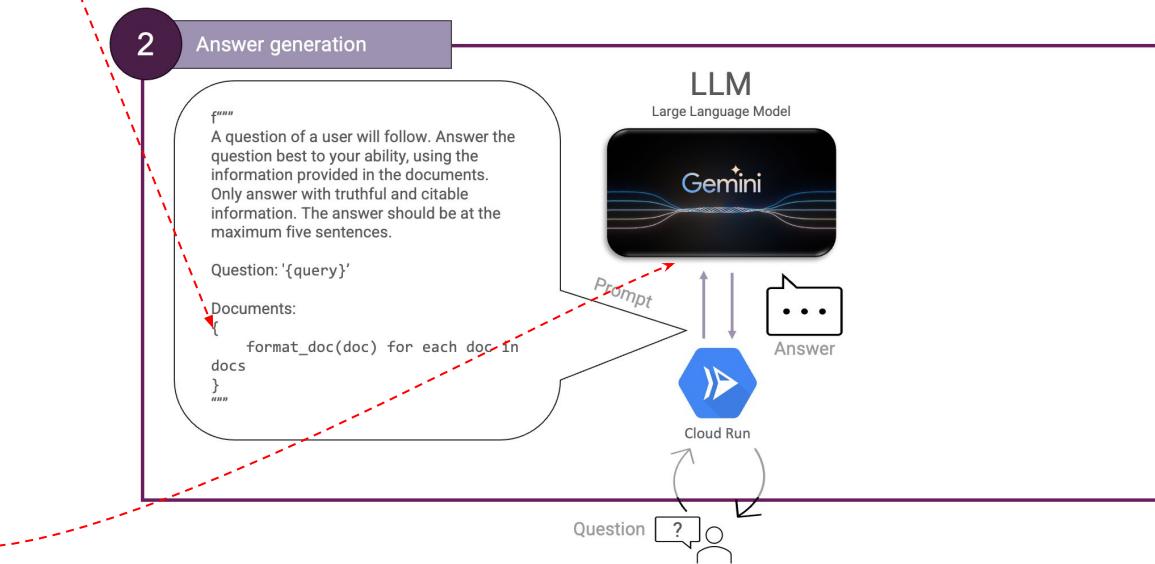
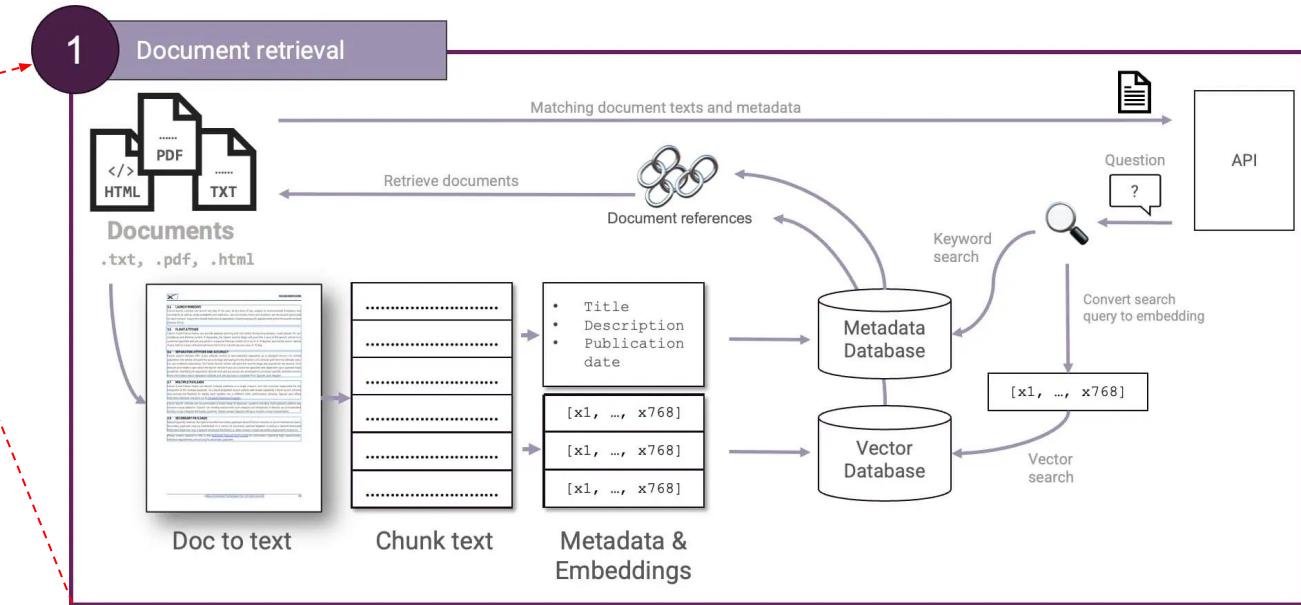
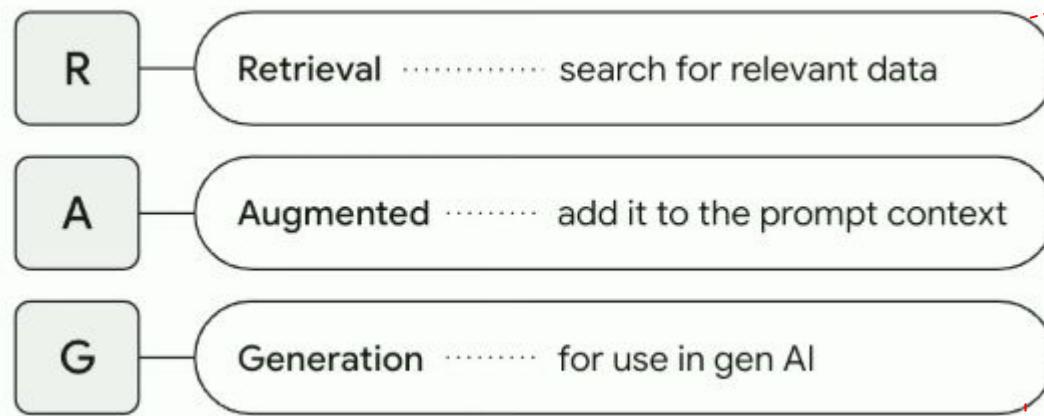


## Prompt and context augmentation

- zero-shot vs (one)few-shot
- Chain of Thought (CoT)
- **Retrieval Augmented Generation (RAG)**

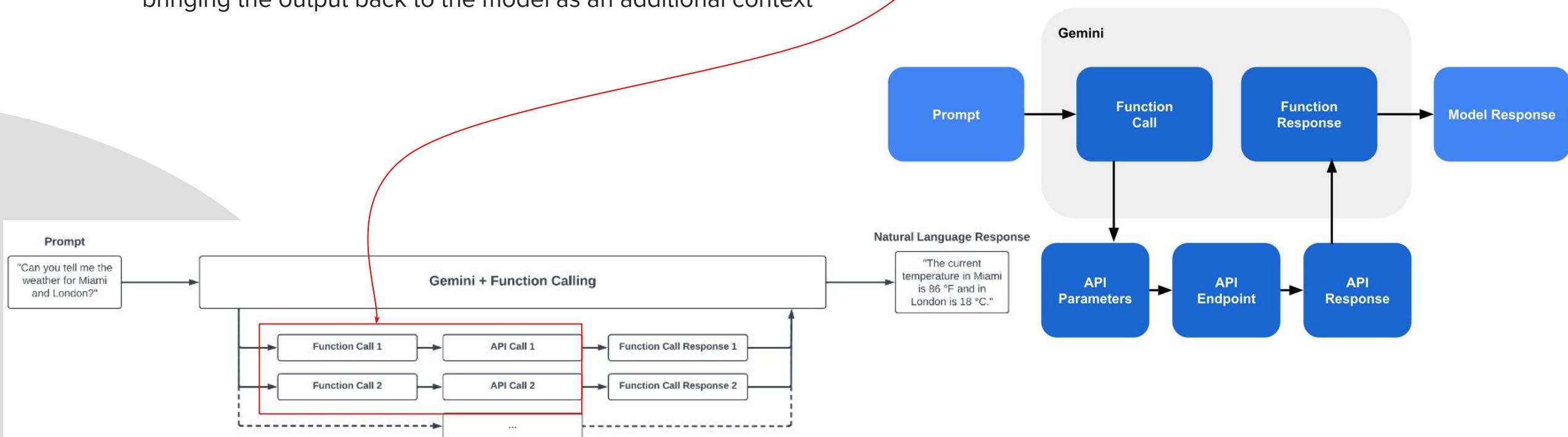


# RAG in 2 steps



# Gemini function calling

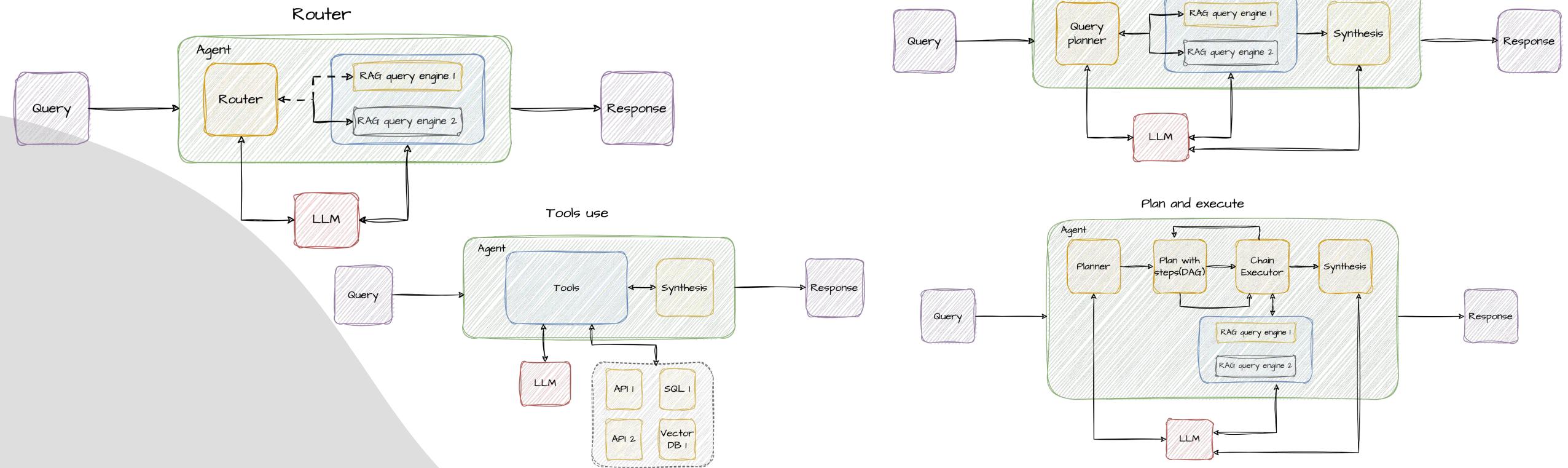
1. define custom functions (e.g. external REST APIs) and provide these to a generative AI model.
2. during processing a query, the model can choose to delegate certain data processing tasks to these functions
3. model does not call these functions just choose them and provides structured input
4. external orchestration mechanism needed for calling and bringing the output back to the model as an additional context



# Towards agentic RAG architecture

Enhanced/generalized set of architectures that combine:

- Agent - an automated ***reasoning*** and decision engine
- Tool/function calling - ***dynamically*** reach out to other knowledge sources
- Reasoning ***loop/iterative*** instead of zero-shot strategy (memory)





# GenAI jump start

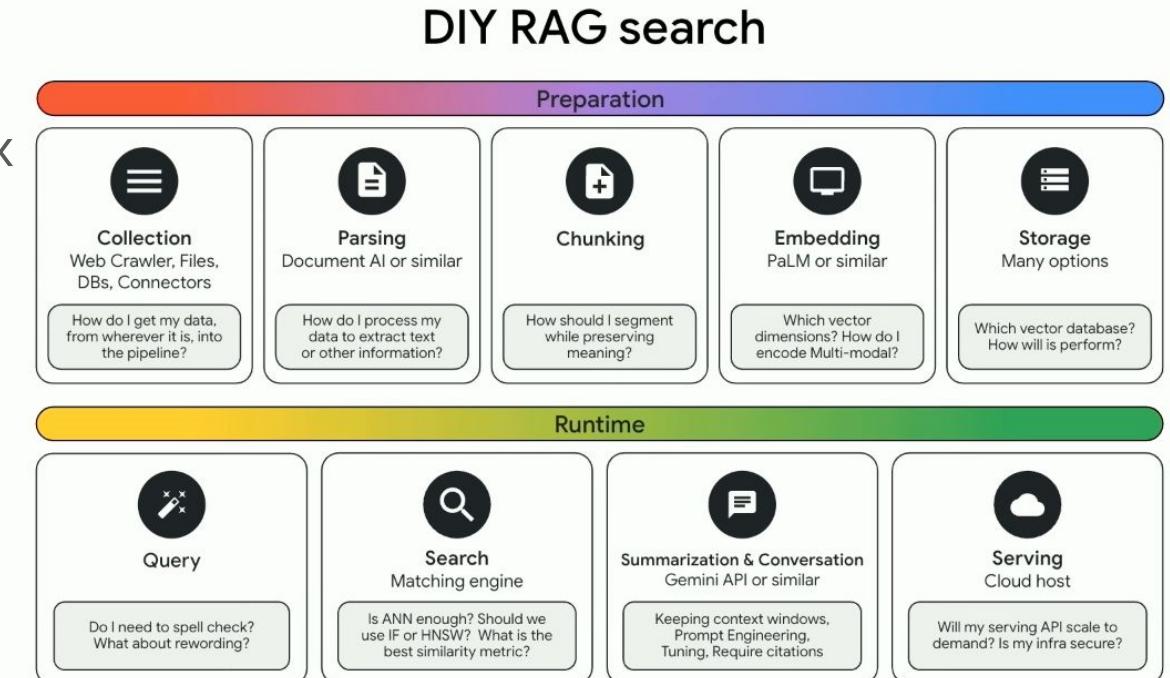
# Generative AI building blocks

- **Vector database** for semantic/hybrid search (knowledge database)
- Large Language Models (for **embeddings** and **text generation**)
- **Integrations:**
  - Data sources ( structured/unstructured )
  - External APIs for function calling
  - Conversational User Interfaces (e.g. Teams, Slack, etc.)
  - Document Storage (e.g. cloud object storage)
  - (Optional) **high-performance framework** for building APIs
- **Orchestration** (data) framework to combine all above

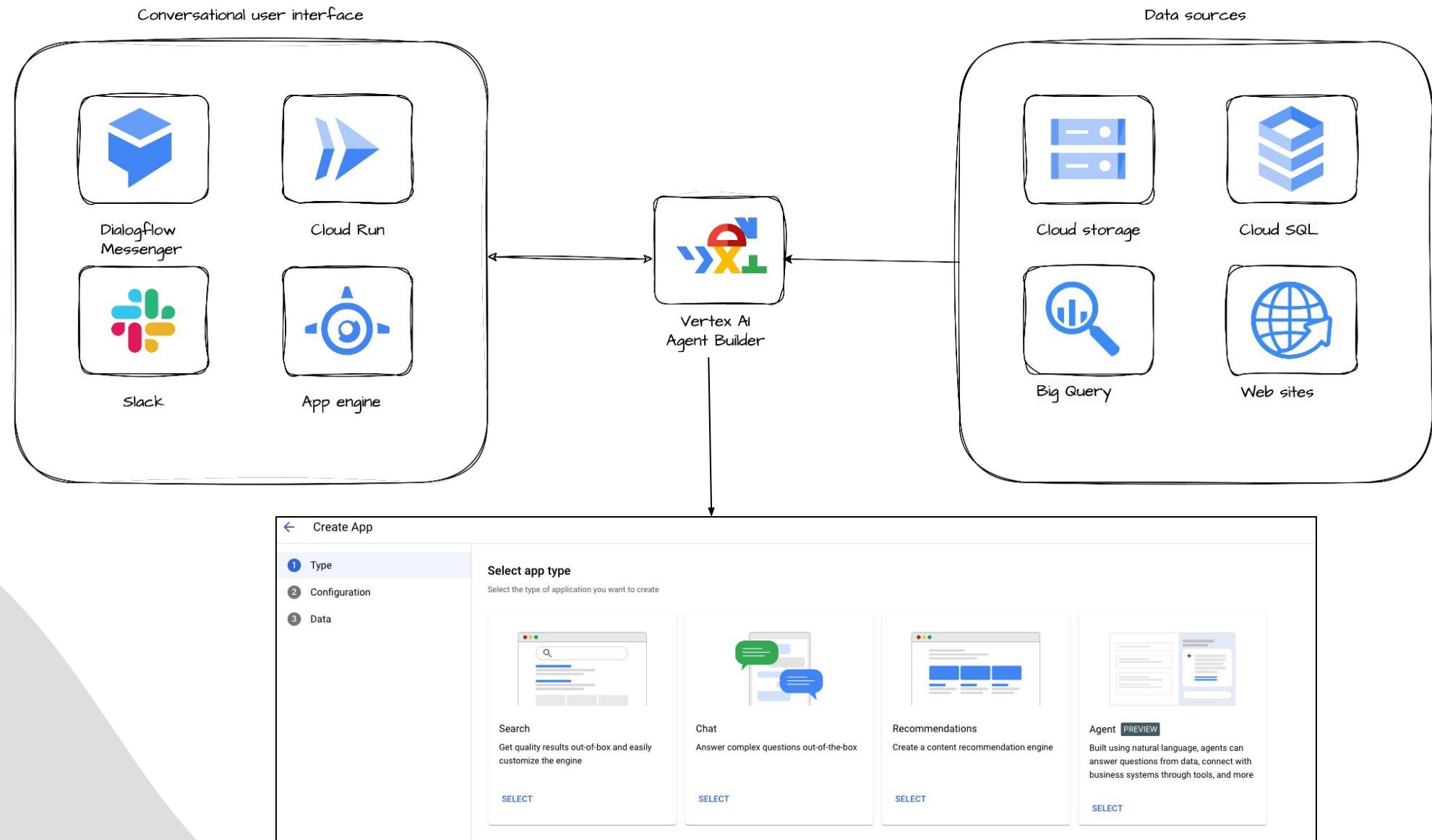


# RAG flavours on GCP - managed vs DIY approach

- *fully managed* with [Vertex AI Agent Builder](#)
- *partly managed* with [Llamaindex](#) and [RAG API](#)
- *full control* - combination of choice  
of open-source (e.g. [LangChain](#), Llamaindex )  
and managed services, such as AlloyDB, Vertex  
AI Search and Gemini models family



# Fully managed with Vertex AI Agent Builder



# Chat app (Vertex AI Conversation)

Select app type

Select the type of application you want to create

- Search
- Chat (selected)
- Recommendations

Create App

Type (selected)

Configuration (selected)

Data

Agent configurations

Configure your agent settings  
You can review the pricing for each feature on our [Pricing Page](#).

Agent configurations

Company name \* xebia

Providing your company name helps the model provide higher-quality responses

Show time zone and language

Your agent name

Agent name \* xebia-data-blog

ID: xebia-data-blog\_1716787108327. It cannot be changed later. [EDIT](#)

Location of your agent

Location type

Region Lower latency within a single region

Multi-region Highest availability across largest area

Multi-region \* global (Global)

CONTINUE CANCEL

Environment: Draft  
Flow: Default Start Flow  
Page: Start Page

What are the RAG on GCP flavours?

The RAG flavours on GCP are:  
Fully managed: using Search & Conversation. Partly managed: managed search using Search & Conversation but manual prompt engineering using Gemini. Full control: manual document processing using Document AI, embedding creation and vector database management using Vertex AI Vector Search.

{ Custom payload }

Talk to agent

Create a Data Store

1 Source (selected)

2 Data

3 Configuration

Select a data source

Choose a data source for your data store

Search sources

Native sources

- Website URLs
- Cloud Storage
- API

Import data from your storage bucket.

Import data manually by calling the API.

SEE DOCUMENTATION

Agent Builder

Data stores > xebia-blog-gs > Data

xebia-blog-gs

Data store ID: xebia-blog-gs\_1716787050502

Type: Unstructured data

Region: global

Language: N/A

Connected apps: xebia-data-blog

Number of documents: 1

Last document import: May 27, 2024, 7:17:41 AM

VIEW DETAILS

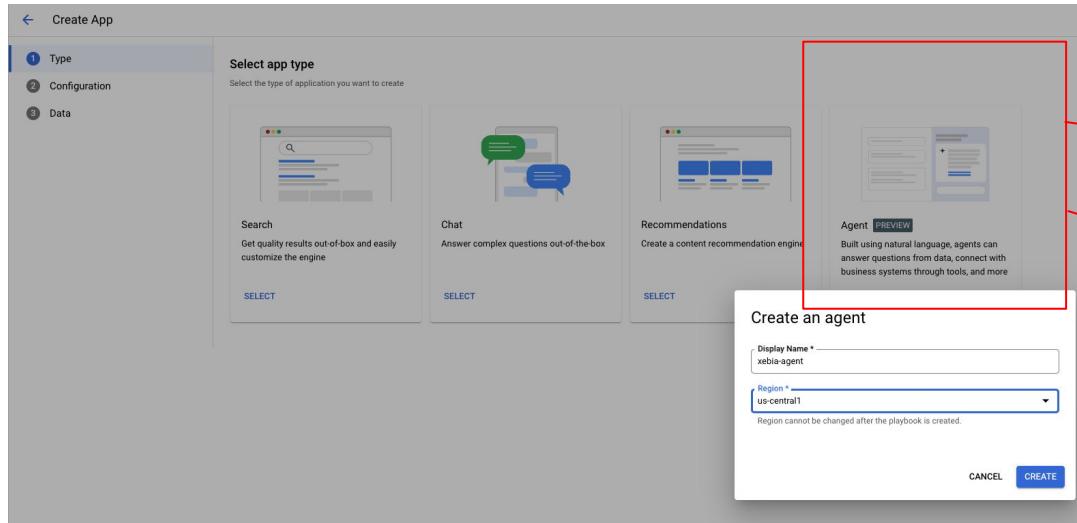
DOCUMENTS ACTIVITY PROCESSING CONFIG PREVIEW

+ IMPORT DATA PURGE DATA

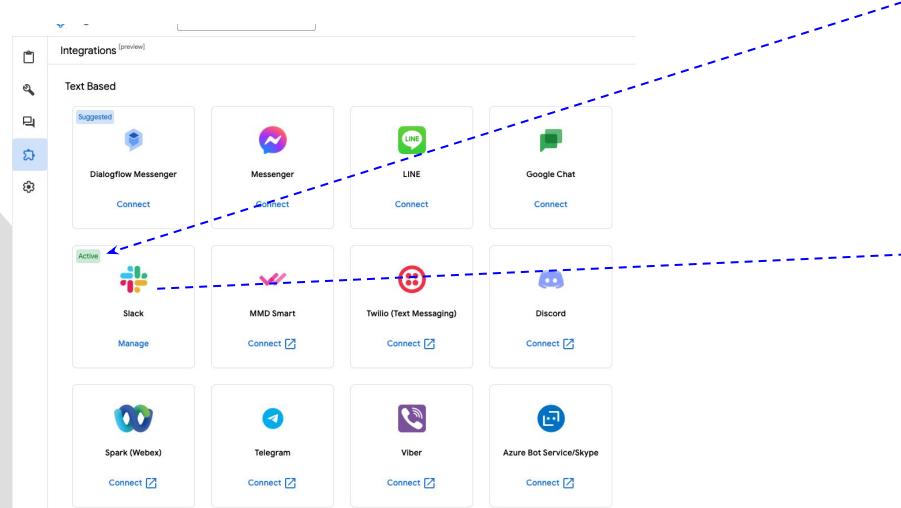
ID: 76596fd02faf647e4c9ddcd05abc325f

URI: gs://xebia-blog-demo/posts/RAG on GCP\_production-ready GenAI on Google Cloud Platform - Xebia.pdf

# Agent App with Slack integration



The screenshot shows the 'Agent Console' for the 'xebia-agent'. In the 'Instructions' section, there is a list of steps: 'Greet user' and 'Use \${TOOL:xebia-blog-gs} to answer user question'. Below this, the 'Available tools' section shows a single tool named 'xebia-blog-gs'. The 'Tool output' section displays a JSON response from the tool, which includes a link to a Google Cloud Platform page about RAG.

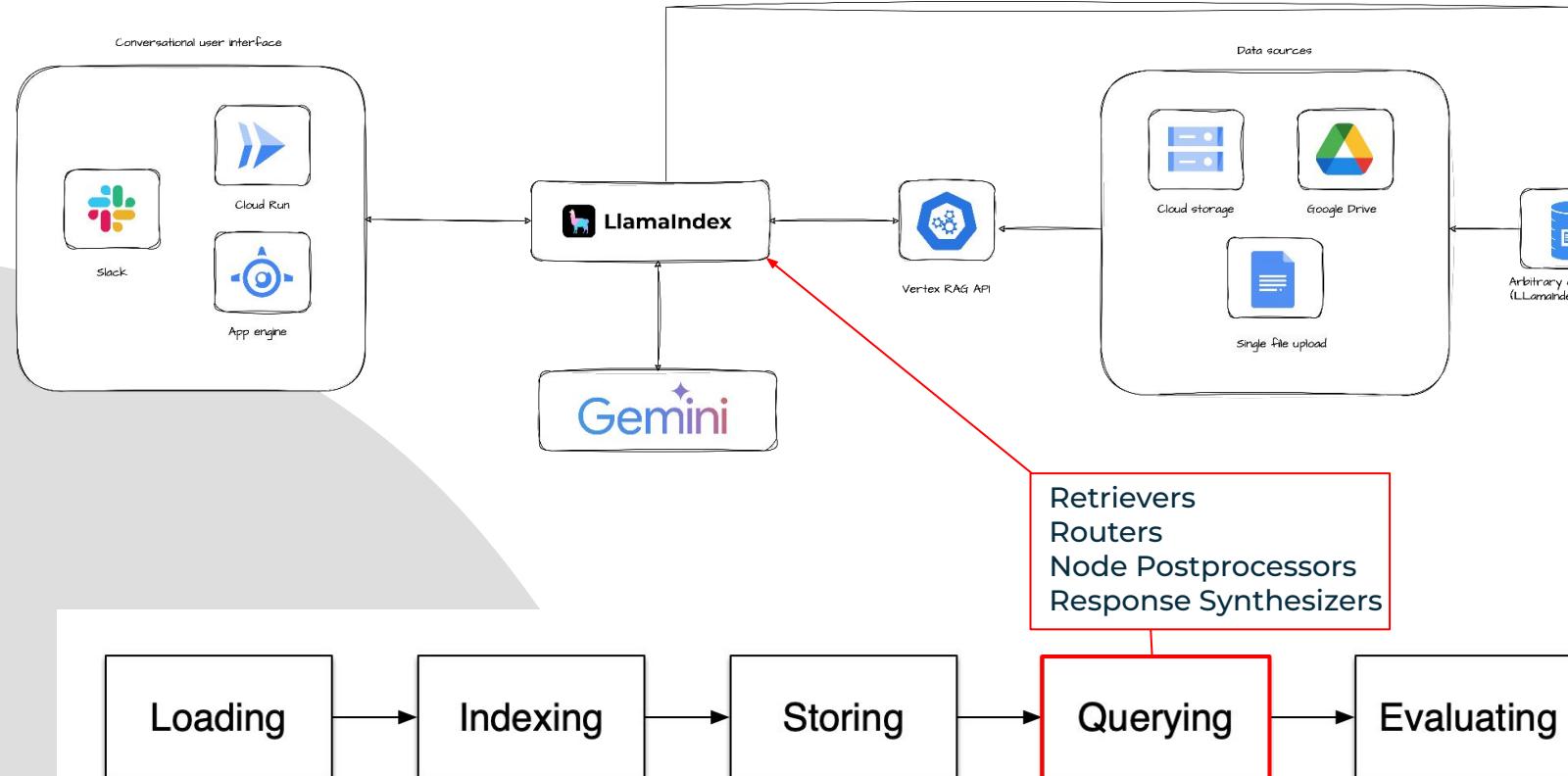


Marek 8:21 AM @VertexAIbot what is RAG ?  
VertexAIbot APP 8:21 AM RAG stands for Retrieval-Augmented Generation. It is a technique that combines document retrieval and answer generation to provide more factual and comprehensive answers. RAG systems use document processors, vector databases, and Large Language Models (LLMs) to retrieve relevant information and generate human-readable answers. Google Cloud Platform offers services that can help build production-ready RAG solutions.

Marek 8:24 AM @VertexAIbot what are RAG flavours on GCP?  
VertexAIbot APP 8:24 AM Google Cloud Platform offers three levels of control in deploying a RAG application: Fully managed: using Search & Conversation. Partly managed: managed search using Search & Conversation but manual prompt-engineering using Gemini. Full control: manual document processing using Document AI, embedding creation and vector database management using Vertex AI Vector Search.

# Partly managed: Llamaindex on Vertex AI

- Existing managed index, retrieval and ingestion APIs in place
- Supported file formats, transformations and data sources
- Llamaindex responsible for orchestration and communication with a model



Announcing Llamaindex on Vertex AI 🎉

We are excited to partner with the Vertex AI team ([Google Cloud](#)) to feature a brand-new RAG API on Vertex, powered by [Llamaindex](#) advanced modules that enable e2e indexing, embedding, retrieval, and generation.

It is simultaneously 🌟 easy to setup and use, while 🦙 providing developers programmatic flexibility to connect a range of data sources (local, GCS, GDrive) and file types (PDF, GDoc, Slides, Markdown) and experiment with both indexing and retrieval parameters.

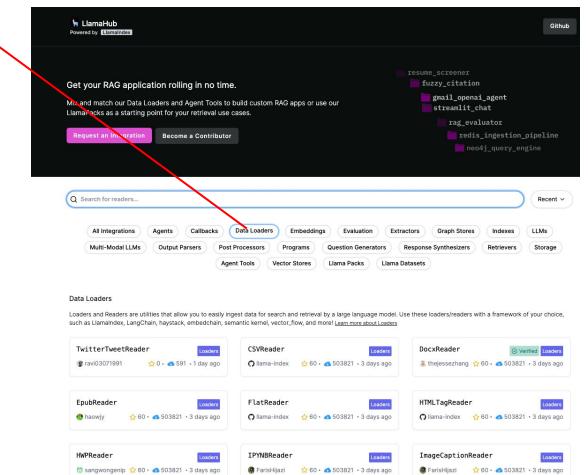
Of course it supports all the latest LLMs:

- ✓ Gemini 1.5 Flash
- ✓ Gemini 1.5 Pro
- ✓ Gemini 1.0 models

Full examples, API reference, and pricing information are provided in the docs below.

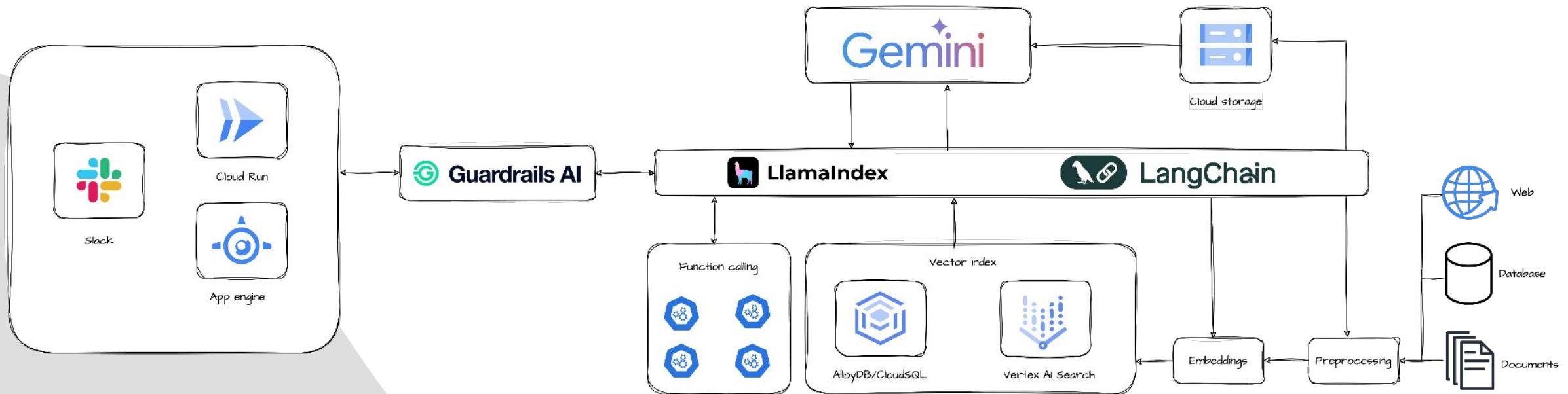
Llamaindex on Vertex AI Docs: <https://lnkd.in/dSK7cQVP>

Vertex I/O announcement blog: <https://lnkd.in/dWShgbAR>



# Full control with [Langchain](#) on Vertex AI

- end-to-end control over
  - ingestion mechanisms
  - embeddings computations
  - underlying vector search engine, such as AlloyDB, Vertex AI Search, CloudSQL with pgVector
  - runtime environment, such as GKE, Cloudrun, on-premise
  - selection of a data framework, such as [Llamaindex](#) or [Langchain](#)





Going live !

# When Going to Production - 4 principles

## IaC

Replace all manual effort with Terraform

## CI/CD

All code is provisioned using CI/CD

## Observability

Every component is monitored (incl. LLMs)

## Security

Infrastructure best practices and LLMs guardrailing



Faster go to market times



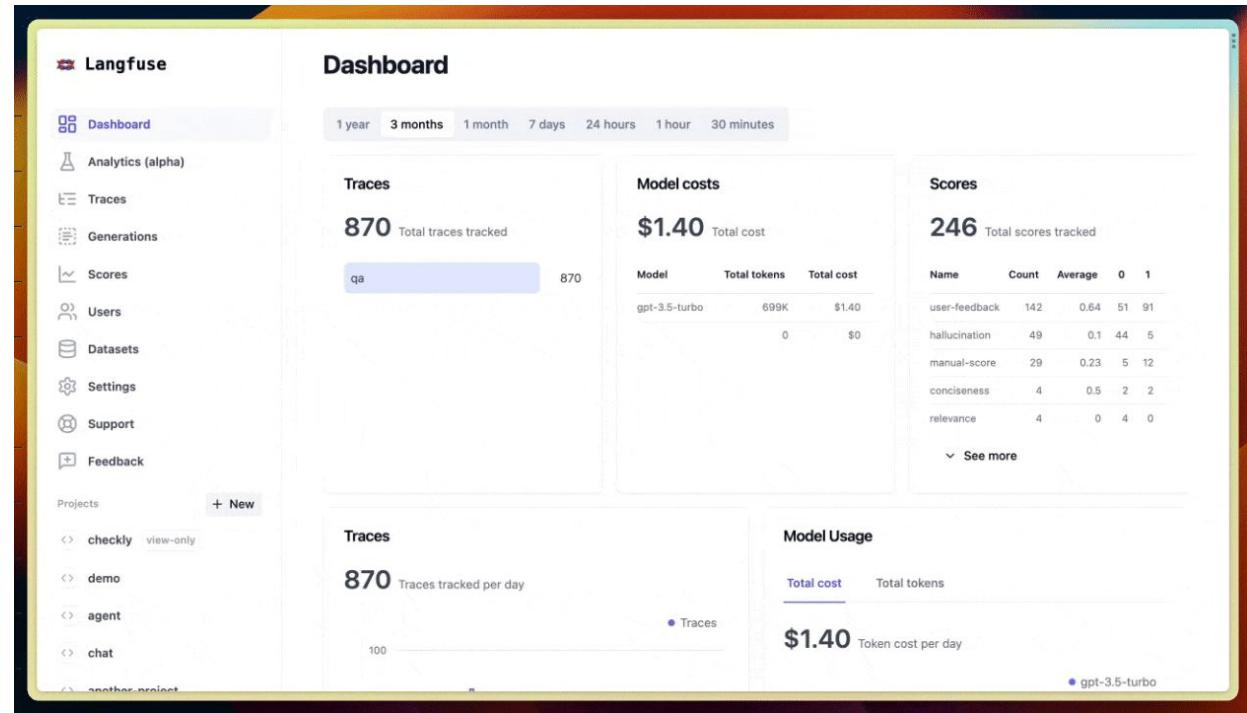
Lower Operational Costs



Higher Auditability

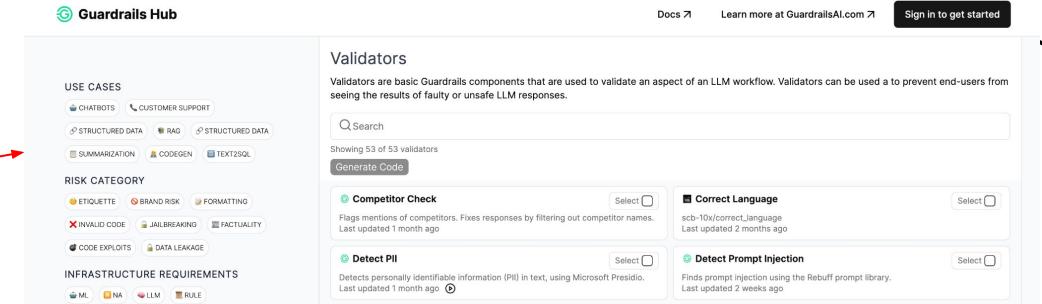
# On LLM monitoring with Langfuse

- observability and tracking platform
- monitor costs, utilization and latency of models
- monitor user interactions with LLMs
- easy integration with Llamaindex or Langchain



✓ *Easily monitor your traces, costs and feedback!*

# LLM guardrailing



- Guardrails a Python framework with a collection of pre-built measures of specific types of risks (called **validators**)
- Presidio is focused on identification and anonymization of private entities in text and images
  - set of different techniques
  - can be deployed also as a service and integrate with llm proxy like litellm
- llm-guard provides set of additional scanners like secrets, toxicity and other prompt injection attacks
- alternatives: last\_layer, PromptArmor (commercial)

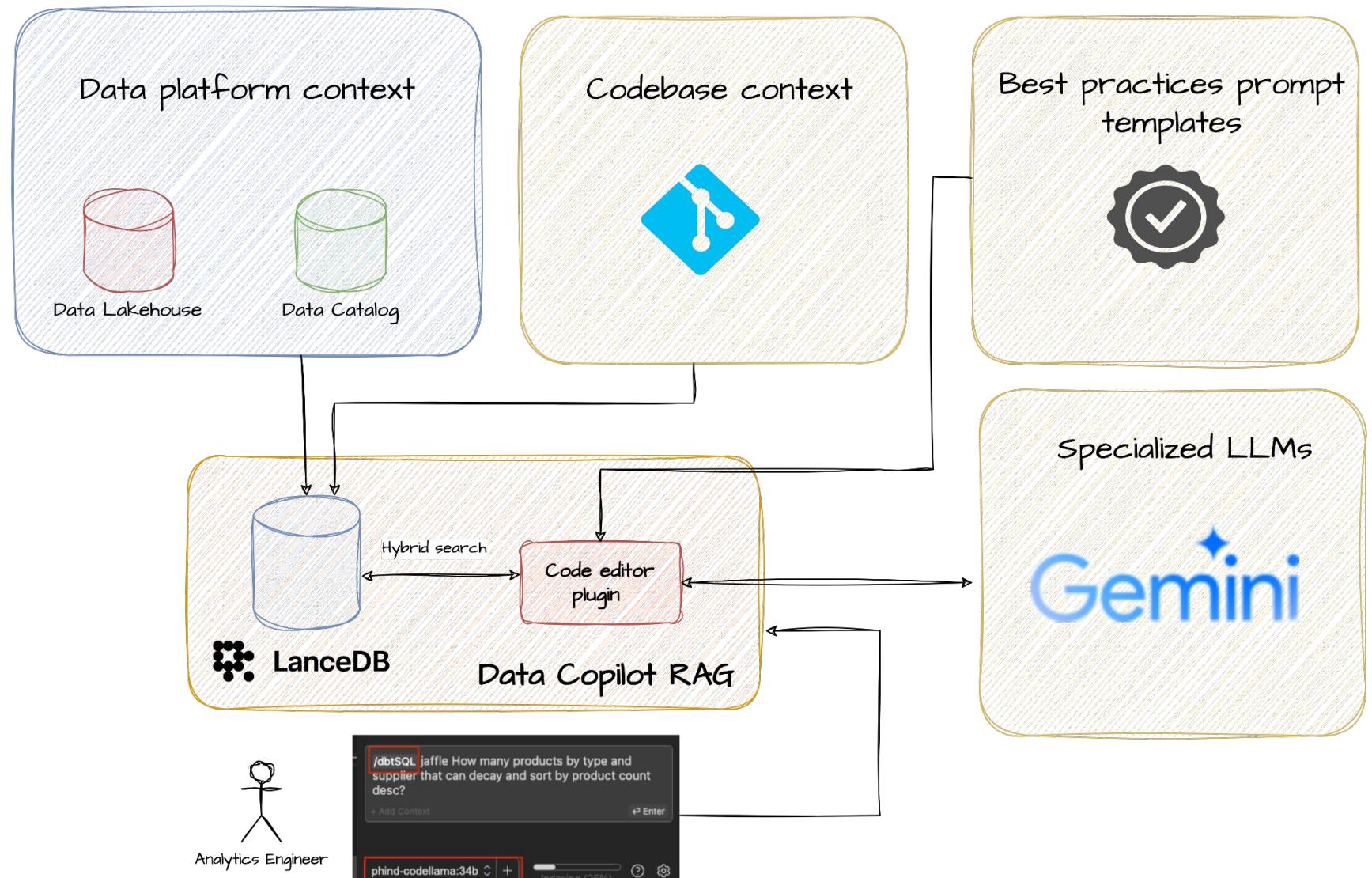
## Presidio Detection Flow





**DEMO time !**

# Xebia Data Copilot architecture



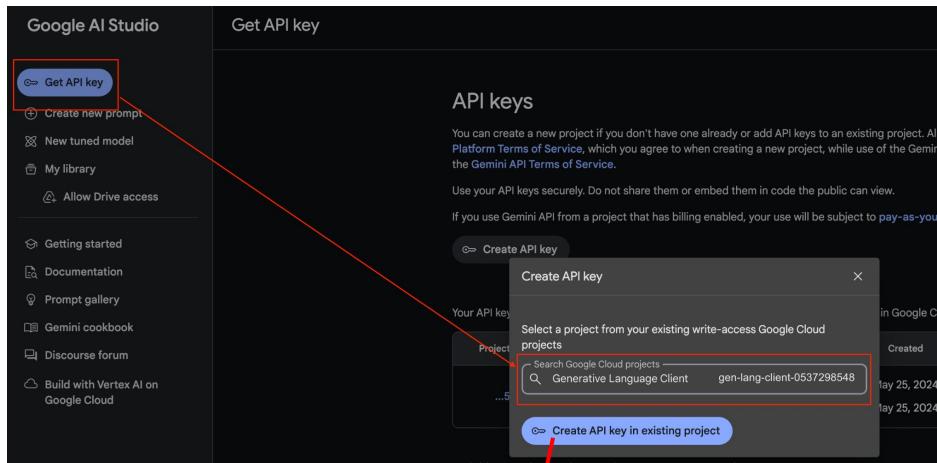
# Xebia Data Copilot Primer

- An extensible AI programming assistant for **SQL** and **dbt code**
- Powered by:
  - Large Language Models (SOTA LLMs)
  - Robust **RAG** architecture
  - **Hybrid** search techniques
  - Fast Vector Database
  - Curated Prompts
  - Builtin Data commands
  - Integrate with VS Code through open-source [Continue.dev](#) copilot

The screenshot illustrates the Xebia Data Copilot extension integrated into VS Code. The interface includes:

- Code Editor:** Displays dbt SQL code for a model named `perishable_products`. The code performs a join between `products` and `supplies` tables to filter perishable supplies.
- Terminal:** Shows the output of running the dbt command `dbt run -s perishable_products`, indicating successful execution of 14 models.
- Sidebar:** Contains an AI-generated context box with the heading "Data context with RAG". It asks about finding products that can decay and provides information from the `supplies` table regarding the `is_perishable_supply` column.
- Bottom Bar:** Includes a "Hybrid search" button, a "Data "slash" commands" button, and a "New Session" button.

# Play with Gemini models and Xebia Data Copilot in 3 steps



The screenshot shows a code editor with a file named 'config.json'. The code defines two slash commands:

```
{}
config.json 4 ×

Users > mwiewior > .continue > {} config.json > [ ] slashCommands

176  {
177    "title": "Gemini Pro",
178    "provider": "gemini",
179    "model": "gemini-1.5-pro-latest",
180    "apiKey": "AIza[REDACTED]"
181  },
182  {
183    "title": "Gemini Flash",
184    "provider": "gemini",
185    "model": "gemini-1.5-flash-latest",
186    "apiKey": "AIza[REDACTED]"
187 }
```

The screenshot shows a code editor with several tabs and panels. The main panel displays a dbtSQL query:

```
/dbtSQL jaffle How many products by type and supplier name that can decay and sort by product count desc?

SELECT p.product_type, s.supply_name,
FROM products p
JOIN supplies s ON p.product_id = s.product_id
WHERE s.is_perishable_supply = TRUE
GROUP BY p.product_type, s.supply_name
ORDER BY product_count DESC;
```

To the right, there's a sidebar with a 'dbtModel' section containing:

```
/dbtModel
```

Below the code editor, the terminal shows the execution of the dbt command:

```
PROBLEMS OUTPUT TERMINAL ... zsh + ×
ducts
04:54:07 Running with dbt=1.7.8
04:54:07 Registered adapter: duckdb=1.7.2
04:54:07 Found 14 models, 6 seeds, 20 tests, 6 sources, 0 exposures, 13 metrics, 773 macros, 0 groups, 6 semantic models
04:54:07
04:54:08 Concurrency: 1 threads (target='dev')
04:54:08
04:54:08 1 of 1 START sql table model main.perishable_products .....
..... [RUN]
04:54:08 1 of 1 OK created sql table model main.perishable_products .....
..... [OK in 0.15s]
04:54:08
04:54:08 Finished running 1 table model in 0 hours 0 minutes and 0.45 seconds
(0.45s).
04:54:08
04:54:08 Completed successfully
04:54:08
04:54:08 Done. PASS=1 WARN=0 ERROR=0 SKIP=0 TOTAL=1
(base) ➜ lakehouse-copilot-demo git:(duckdb-setup) x
```

The bottom right corner shows a 'Gemini Pro' status indicator.

# Thank you!

## Marek Wiewiórka



### Copyright Notice

Copyright © Xebia Nederland B.V., Laapersveld 27, 1213 VB, Hilversum, The Netherlands. All rights reserved.  
Xebia® is a registered trademark of Xebia Holding B.V. internationally. All other company references  
may be trademarks and/or service marks of their respective owners.