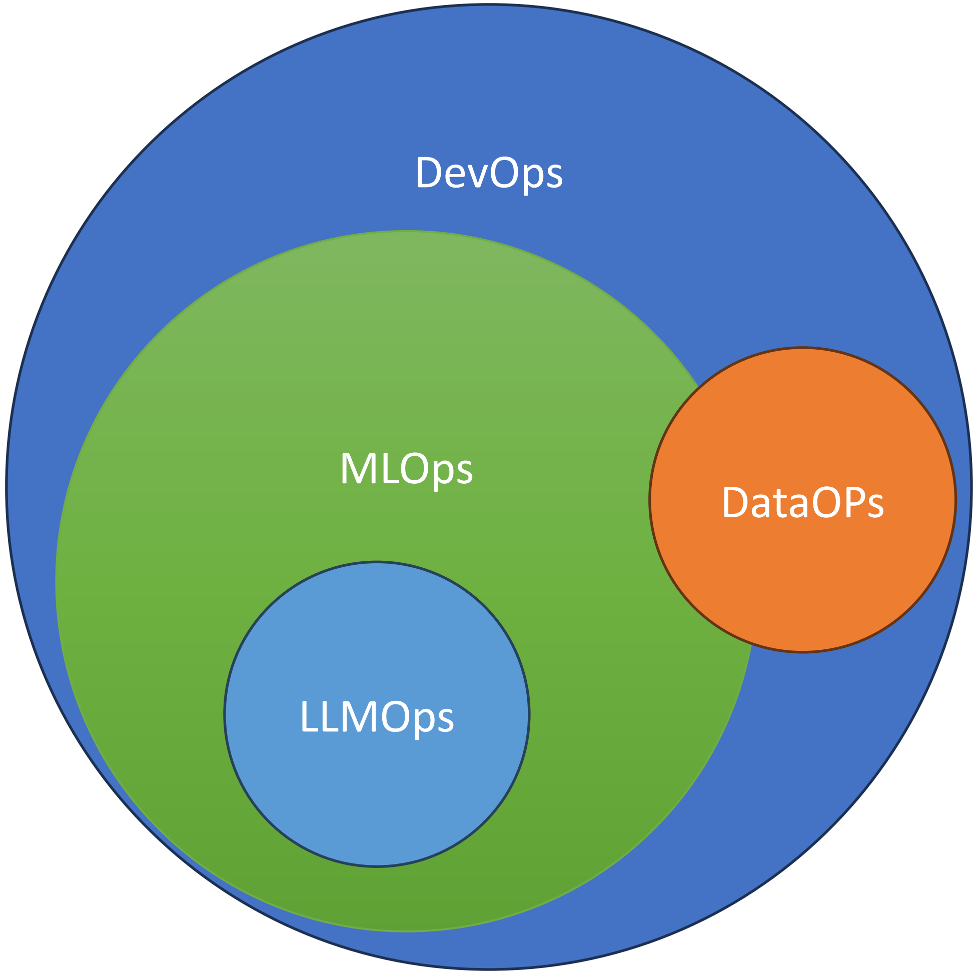
Large Language Models Operations (LLMOps)

Content

- LLMOPs introduction
- Deployment and scalability of LLMs
- Monitoring and maintenance of models in production
- Performance evaluation and continuous improvement
- Ethical considerations and privacy

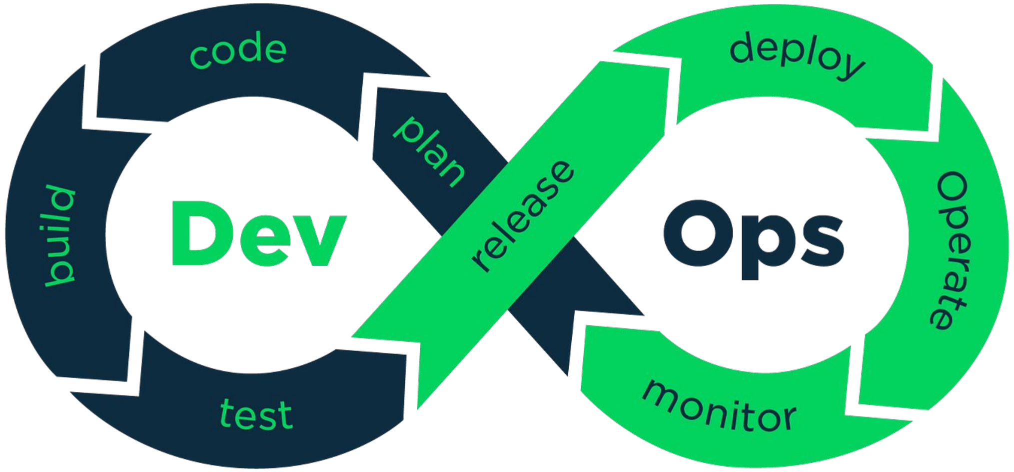
Intro to LLMOPS

XOPs

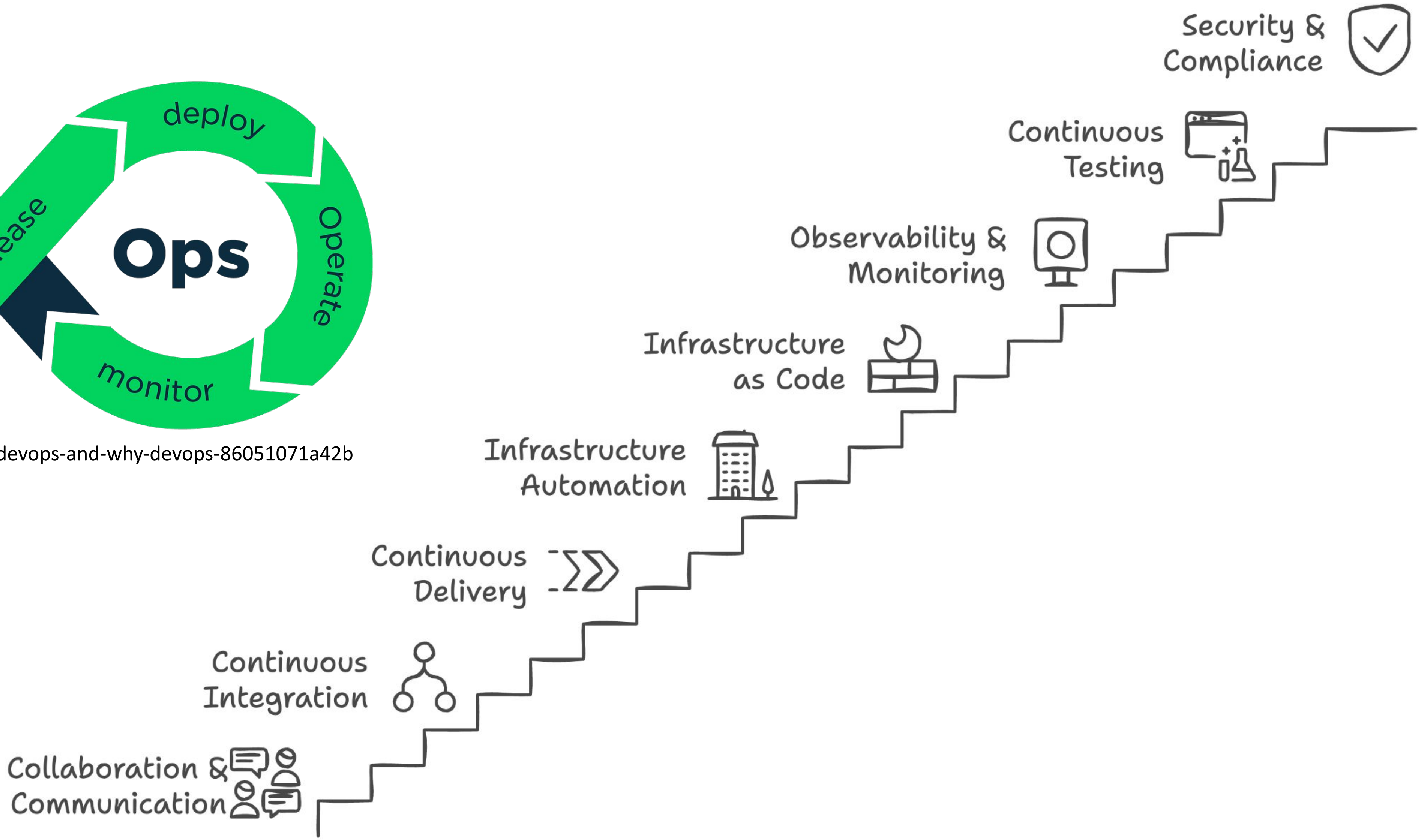


- Automation
- Collaboration
- Reproducibility
- CI/CD
- Observability

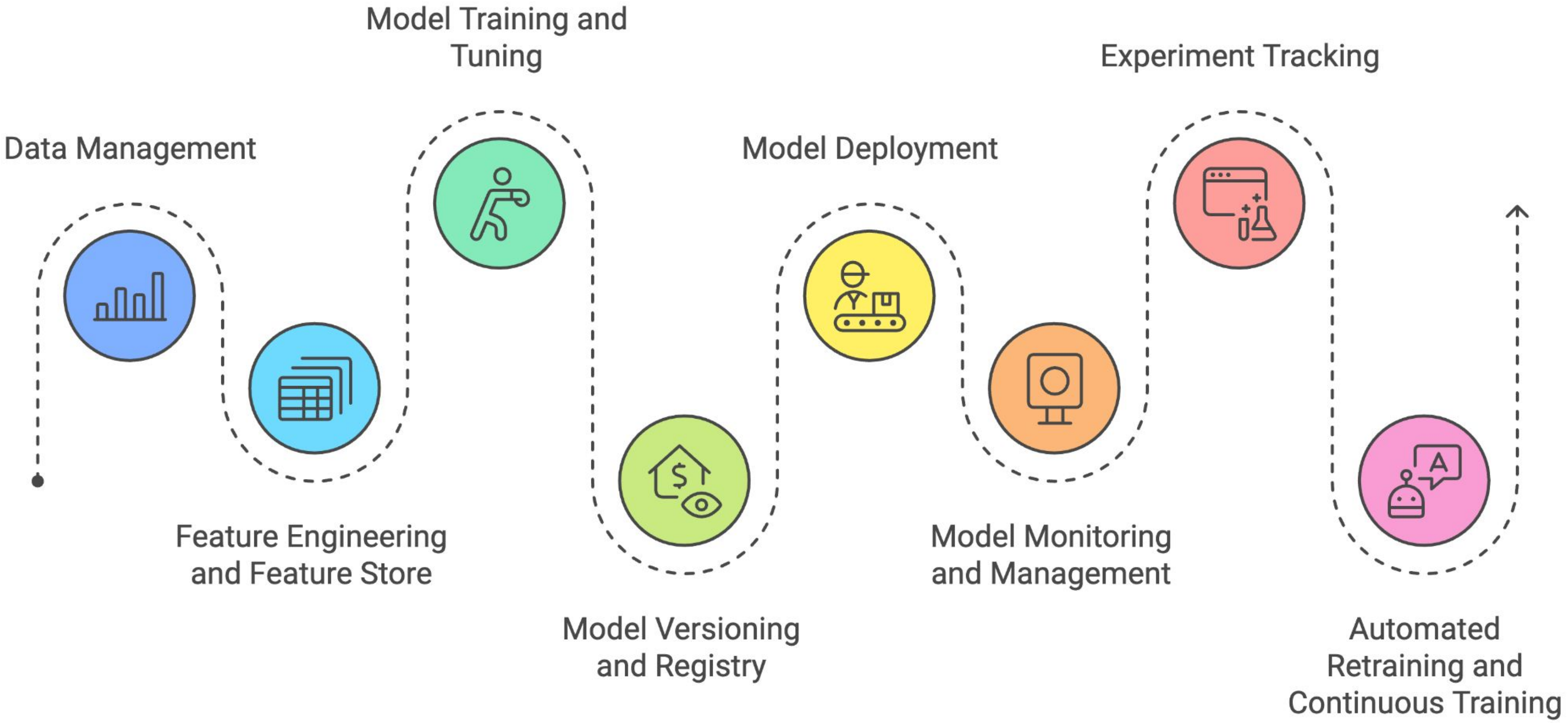
DevOps



https://medium.com/@ritusherke86/what-is-devops-and-why-devops-86051071a42b



MLOps



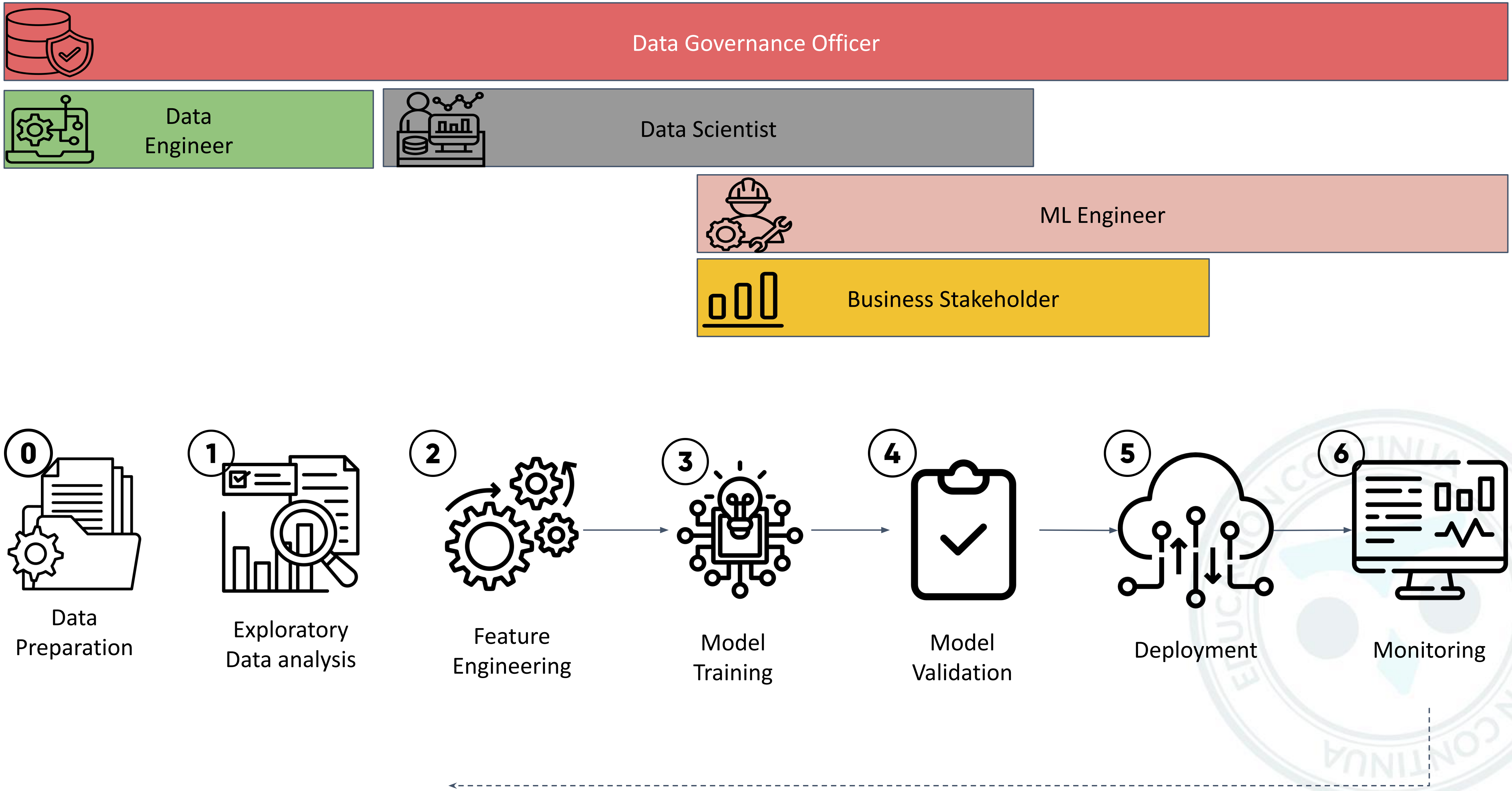
Data Management
Feature Engineering and Feature Store
Model Training and Tuning
Model Versioning and Registry
Model Deployment
Model Monitoring and Management
Experiment Tracking
Automated Retraining and Continuous Training

LLMOps

Resource Optimization
Prompt Engineering
Bias Detection and Mitigation
Database Management
Human Feedback Integration
LLM Architecture
LLM Evaluation Metrics

Training and deployment of Large Language Models

MLOps


Data Governance Officer
Data Engineer
Data Scientist
ML Engineer
Business Stakeholder
0
Data Preparation
1
Exploratory Data analysis
2
Feature Engineering
3
Model Training
4
Model Validation
5
Deployment
6
Monitoring

MLOps

What specific scaling challenges exist for LLMs?

Initial training: trillions of tokens, hundreds to thousands of GBs & very long run times.

Fine-tuning: updating model weights based on your own data, still requires relatively large data and long training times. Plus lots of evaluation!

Storage: the models contain billions of parameters, often hundreds of GB. This can be prohibitive for some devices.

Latency: Running inference on these models can be very costly in terms of time.

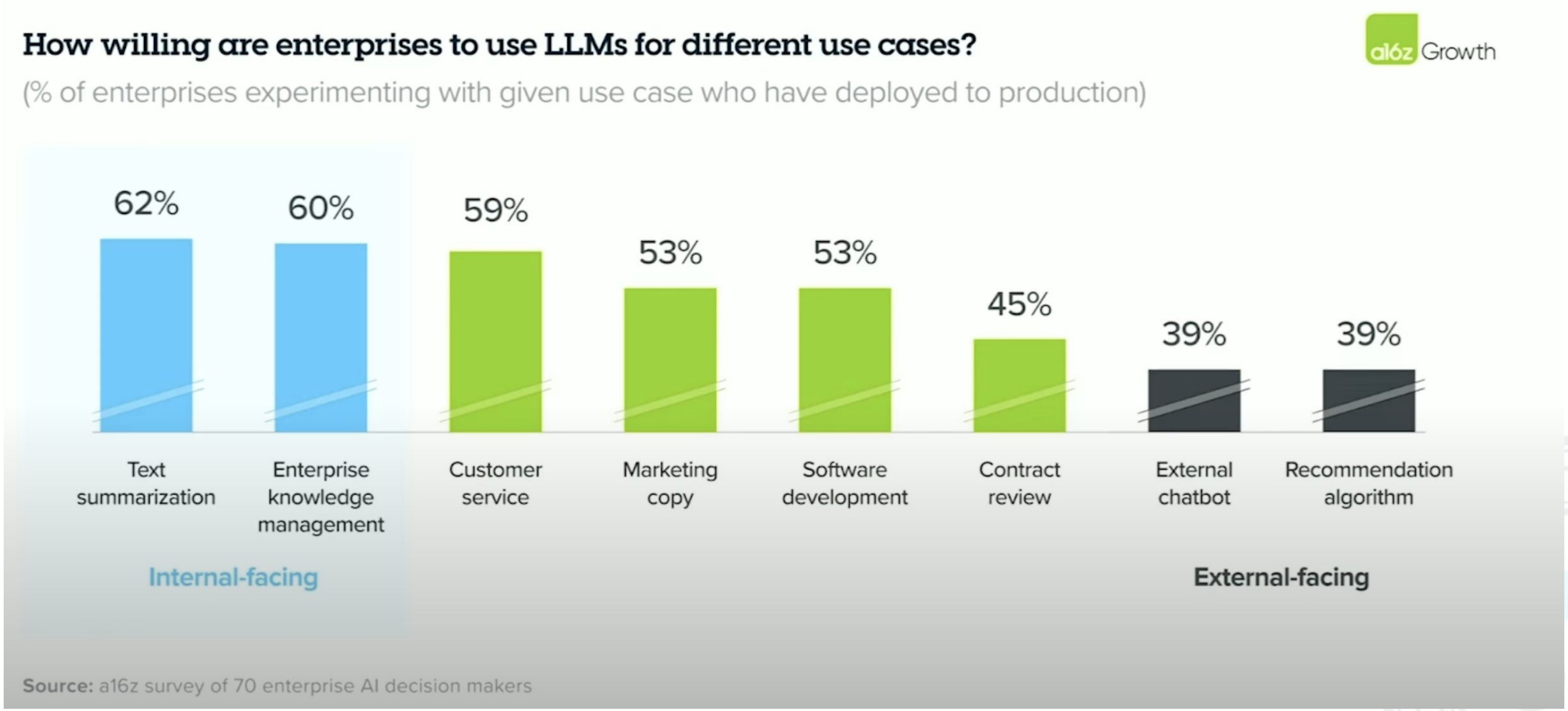
Cost: All of this costs $$$!

MLOps

What specific scaling challenges exist for LLMs?

- **Initial training**: don't do it!
- **Fine-tuning**: Optimize and use a scalable framework, such as Ray.
- **Storage**: Quantization, memorization, caching …
- **Latency**: Quantization, memorization, caching, hardware and memory bandwidth optimization…
- **Cost**: Above plus use 'open source' models

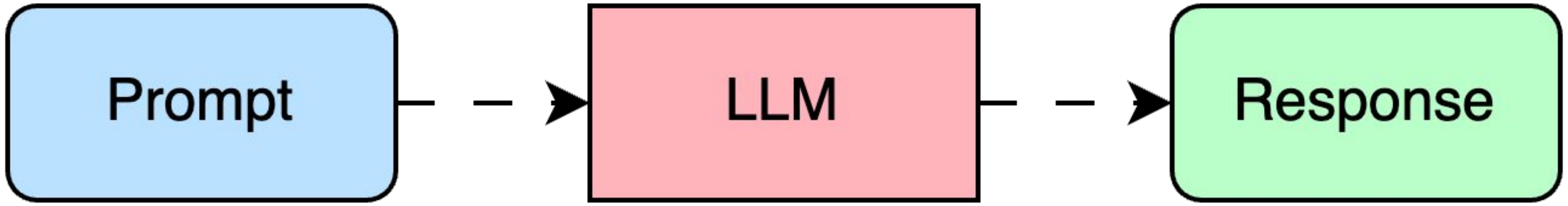
Types of use-cases for Enterprises

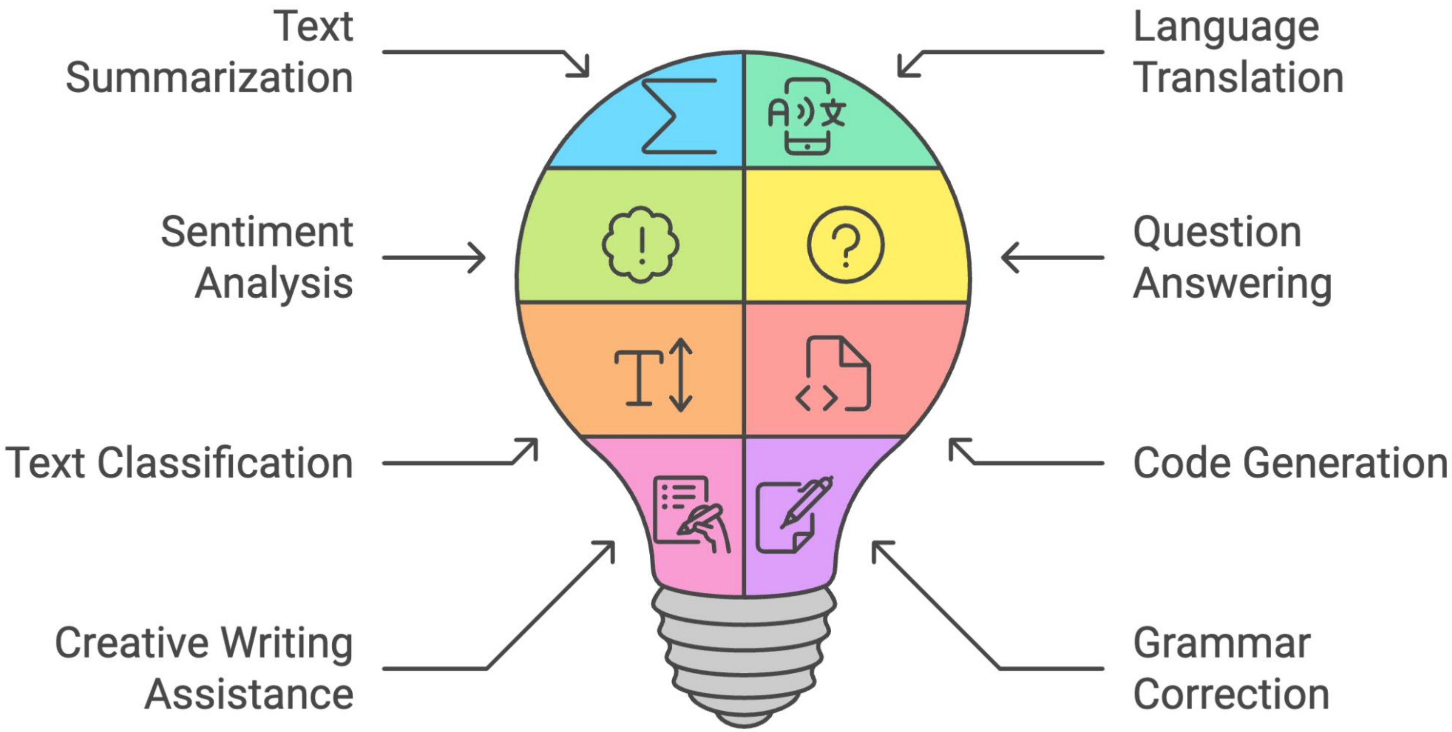


LLM application archetypes

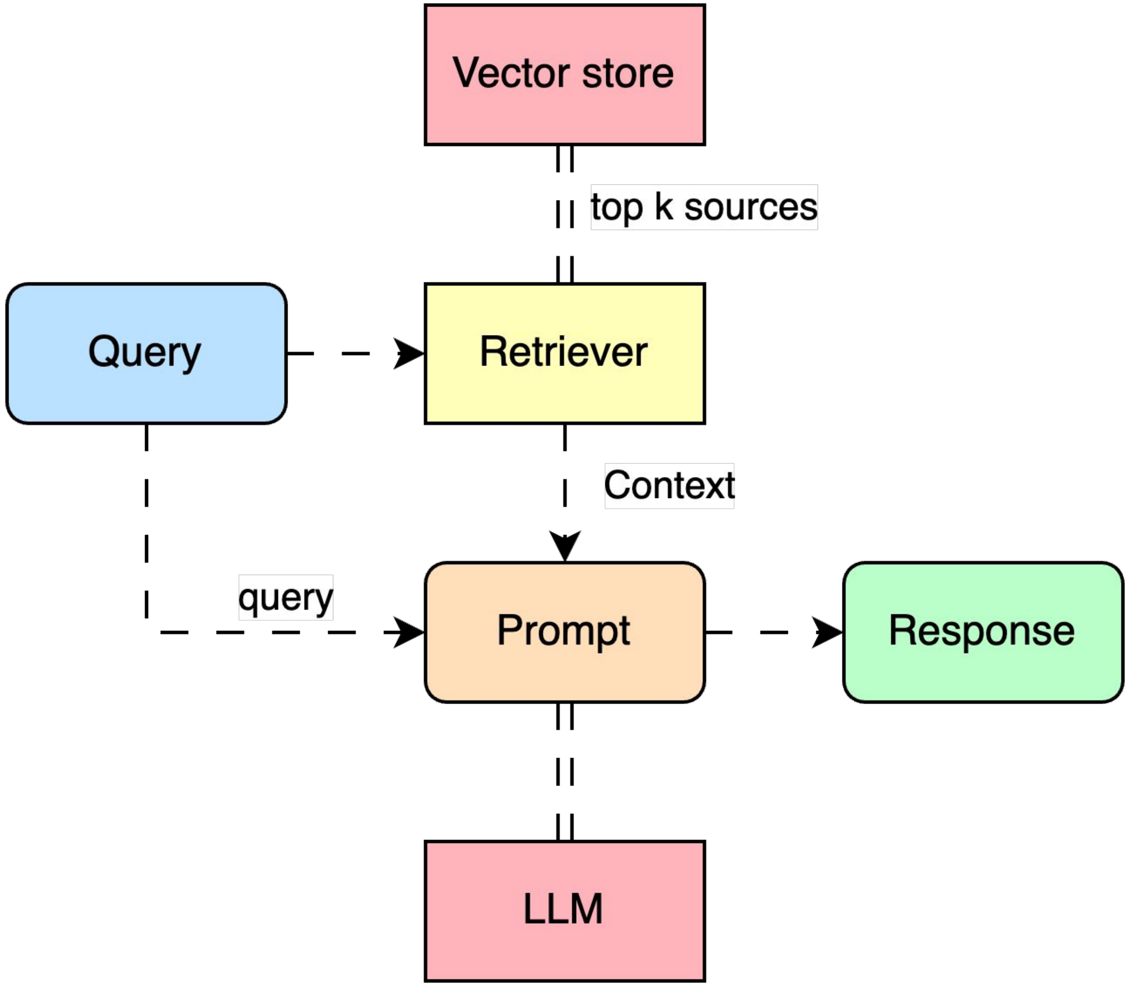
- Prompt Engineering
- Retrieval Augmented Generation
- Agents
- Multi-Agents
- Fine-tunning

Prompt Engineering Applications

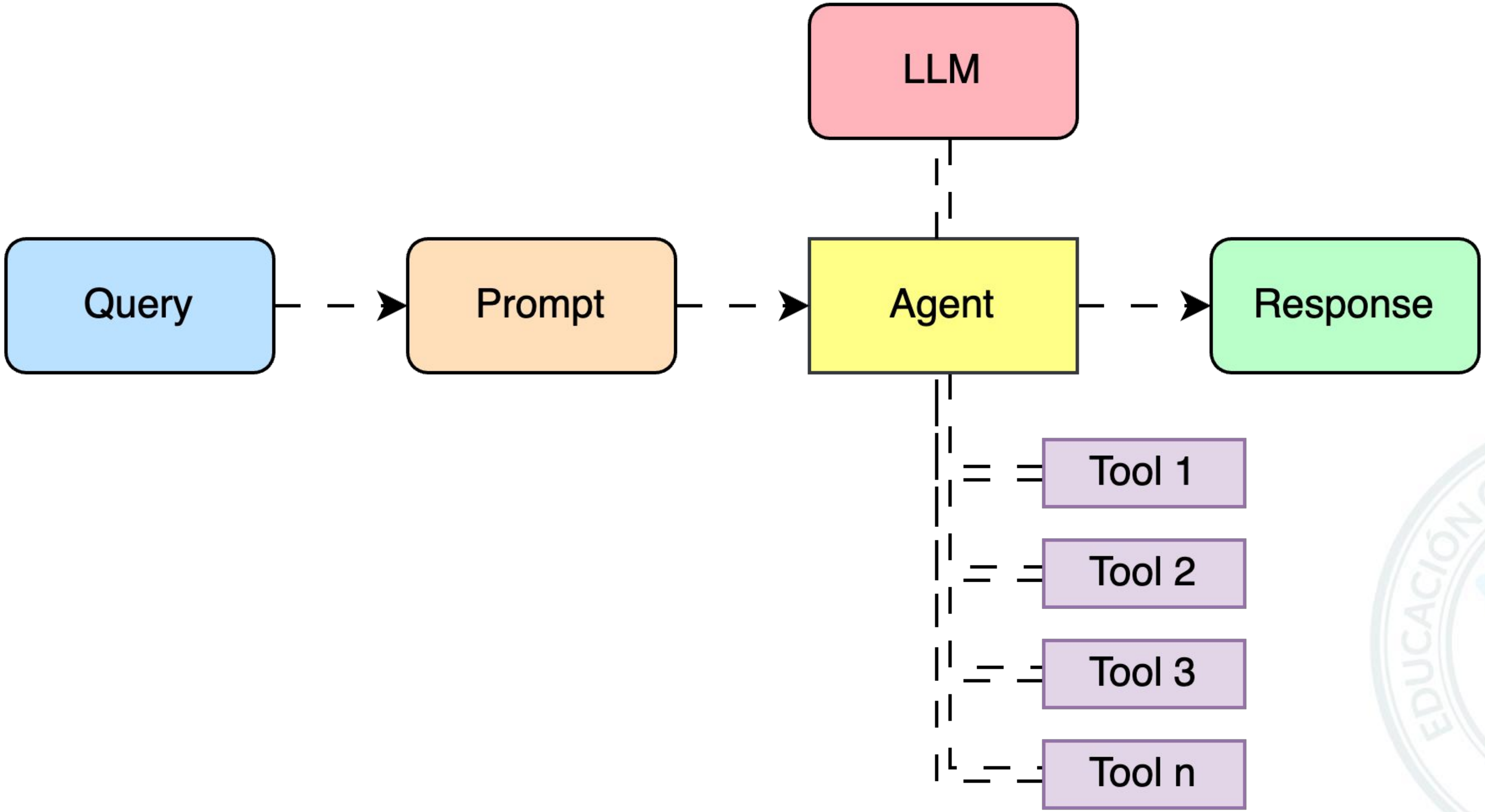




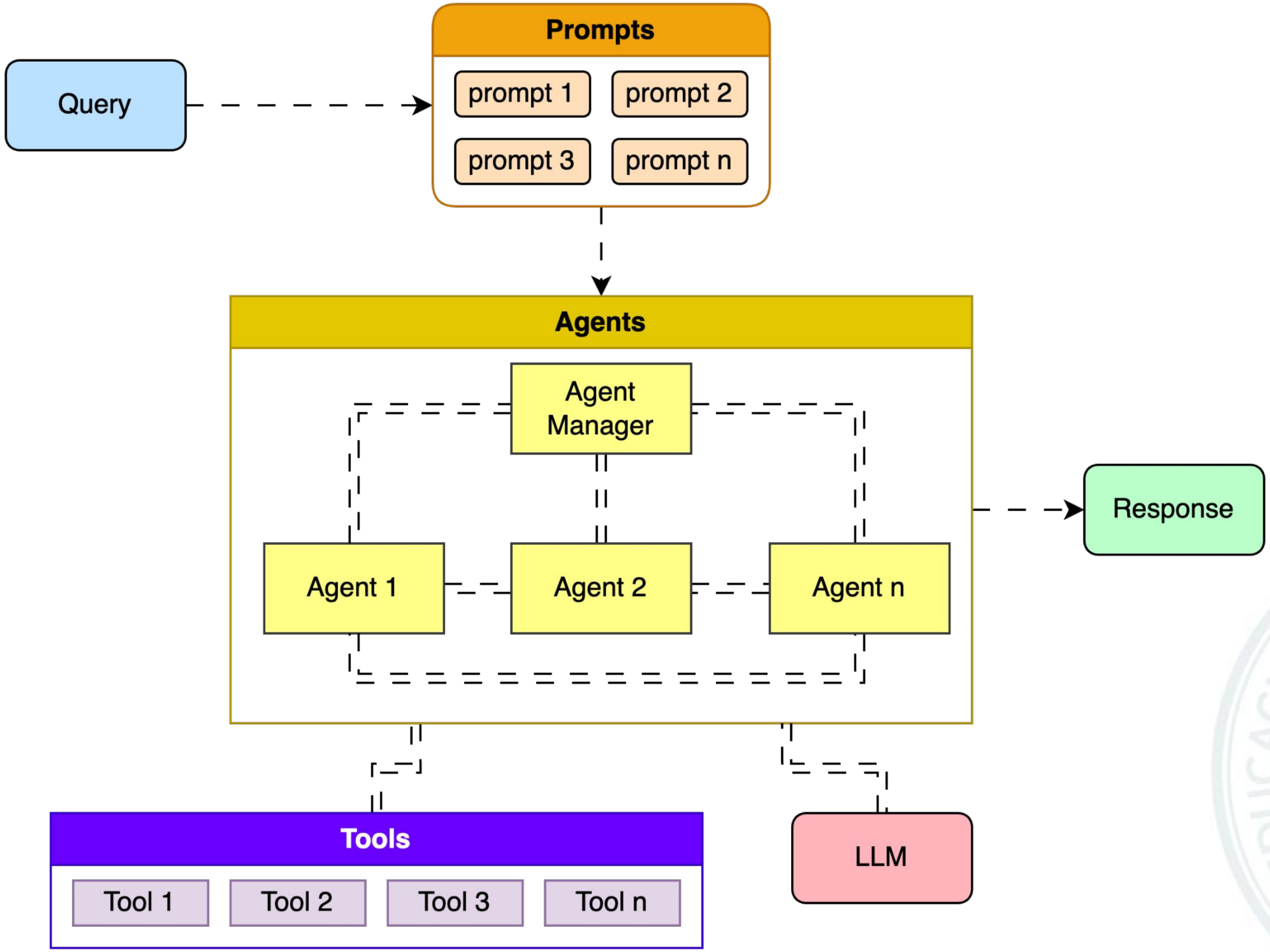
Retrieval Augmented Generation Applications



Agentic Applications



Multi-Agent Applications



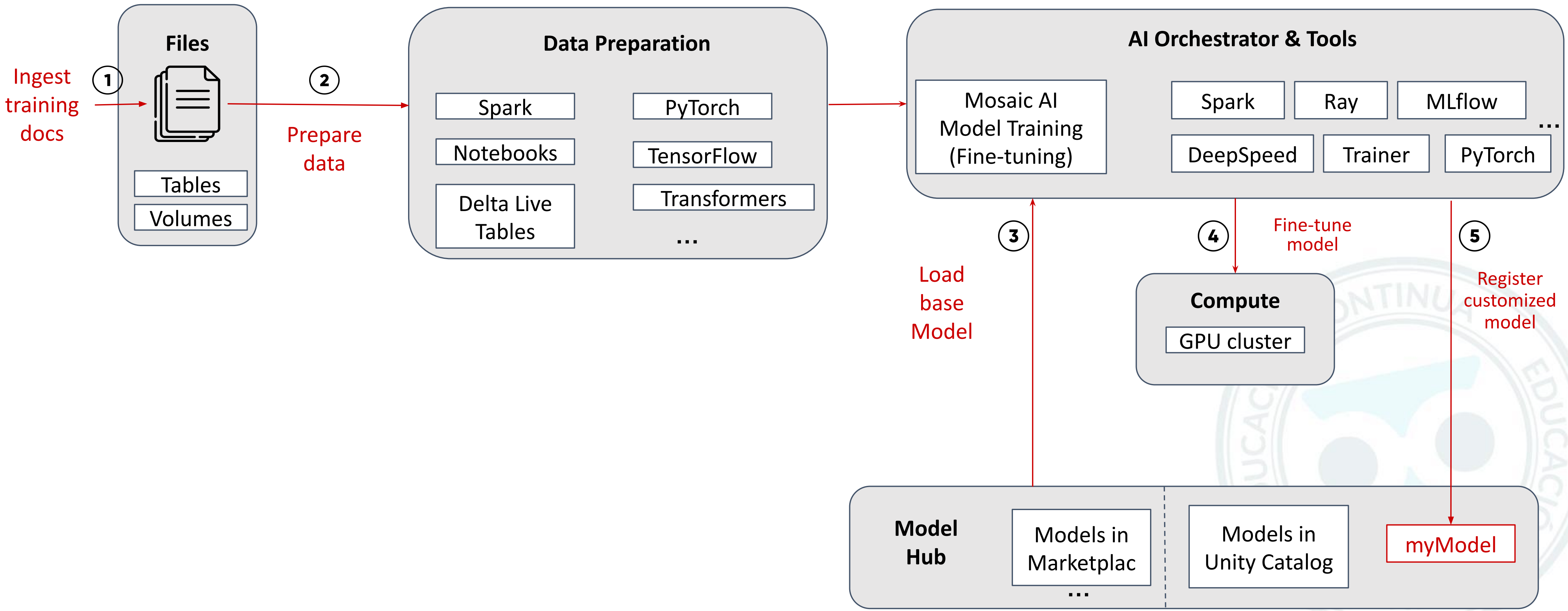
Fine-tunning LLM Applications

Operationalizing Fine-tuning Pipelines

- Automation, version control, reproducibility
- Distributed training infrastructure
  - DeepSpeed, PEFT, GPUs
- Similar to classical model training serving
- Optimization techniques
  - vLLM, MLC, CudaGraph, MQA, Quantization, TensorRT
- Deeper understanding of GPUs, TTFT, TPOT

Types of LLM Applications

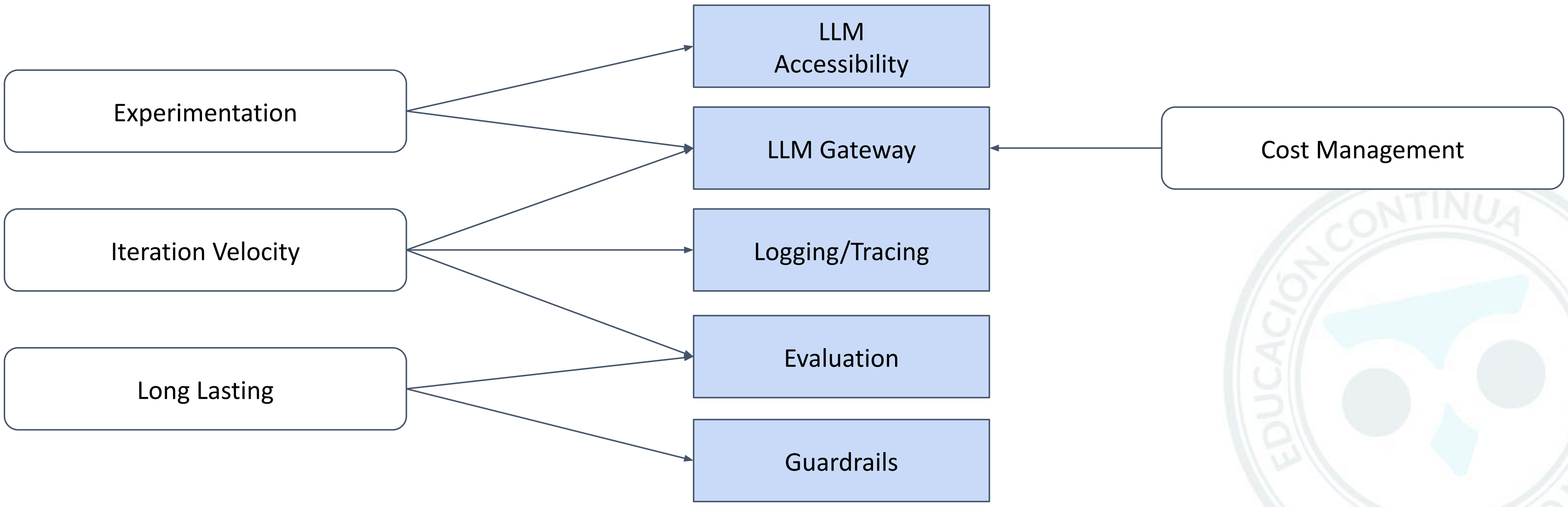
Architecture: fine-tuning



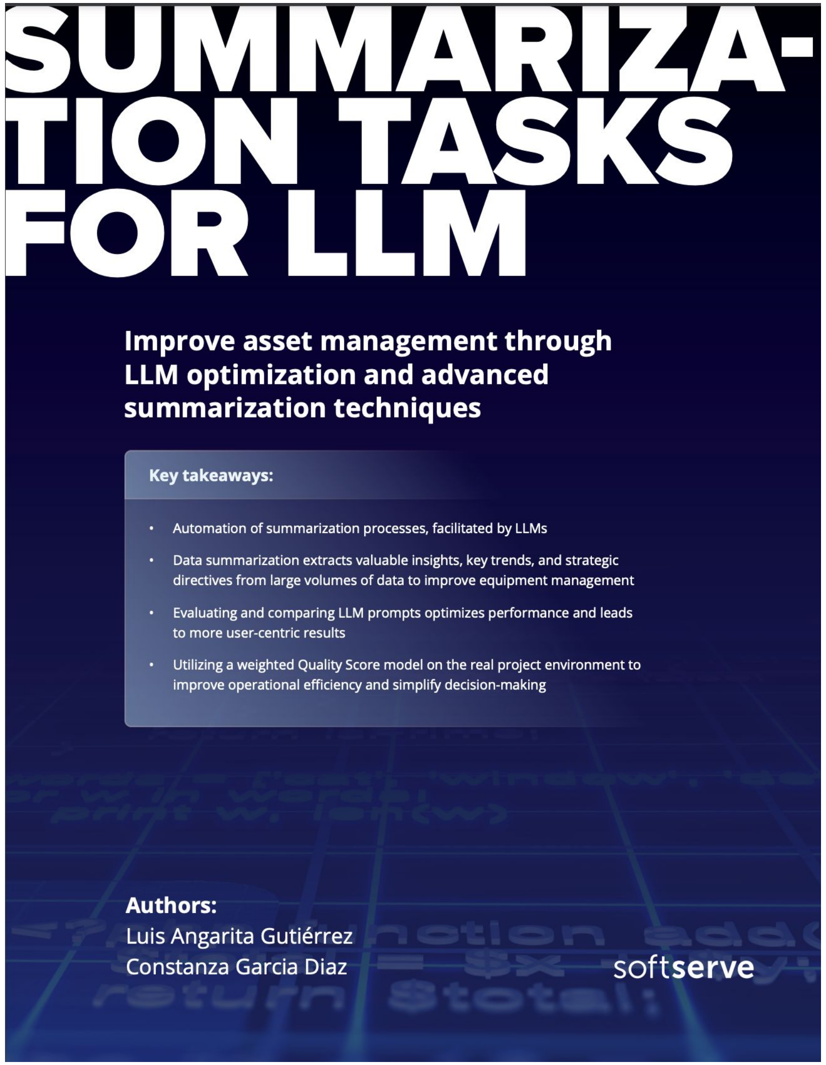
Types of LLM Applications

Practitioner's Perspective

LLMOps Starting Points



Use-Case



SUMMARIZATION TASKS FOR LLM

Improve asset management through LLM optimization and advanced summarization techniques

Key takeaways:

- Automation of summarization processes, facilitated by LLMs
- Data summarization extracts valuable insights, key trends, and strategic directives from large volumes of data to improve equipment management
- Evaluating and comparing LLM prompts optimizes performance and leads to more user-centric results
- Utilizing a weighted Quality Score model on the real project environment to improve operational efficiency and simplify decision-making

Authors:
Luis Angarita Gutiérrez
Constanza Garcia Diaz

softserve

Improve Asset Management with LLM Evaluation Pipelines

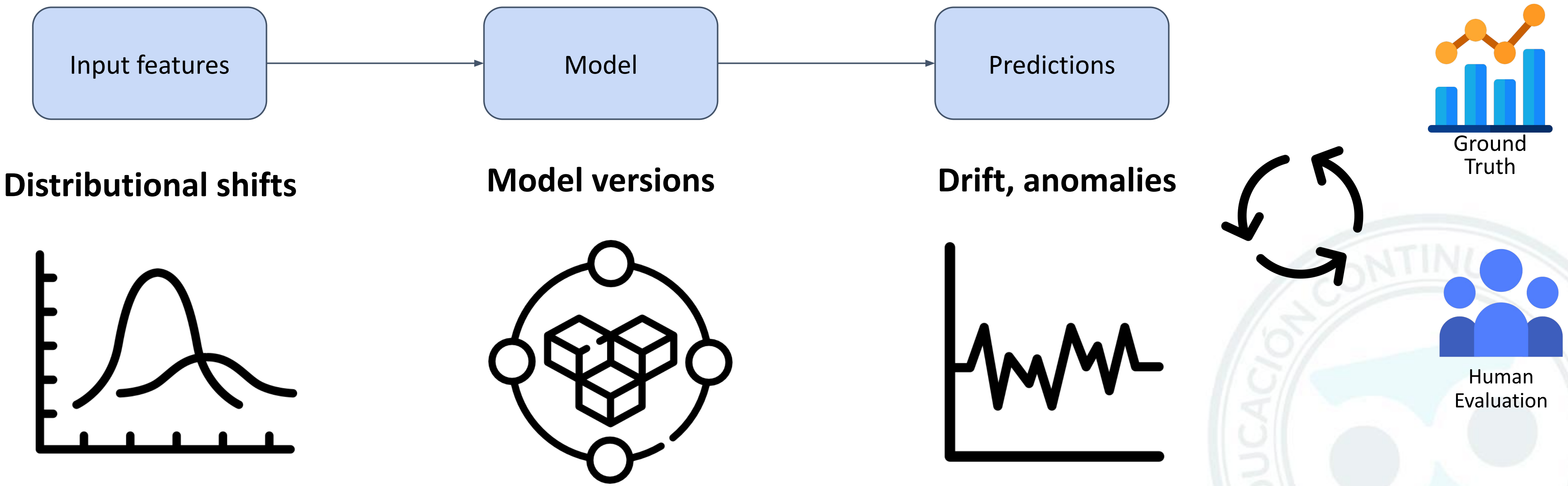
Creating efficient AI-powered summaries, LLMs save up to 93% of the time taken by traditional manual processes. Our white paper explains how this allows you to optimize asset management operations.

https://info.softserveinc.com/summarization-tasks-for-llm-white-paper

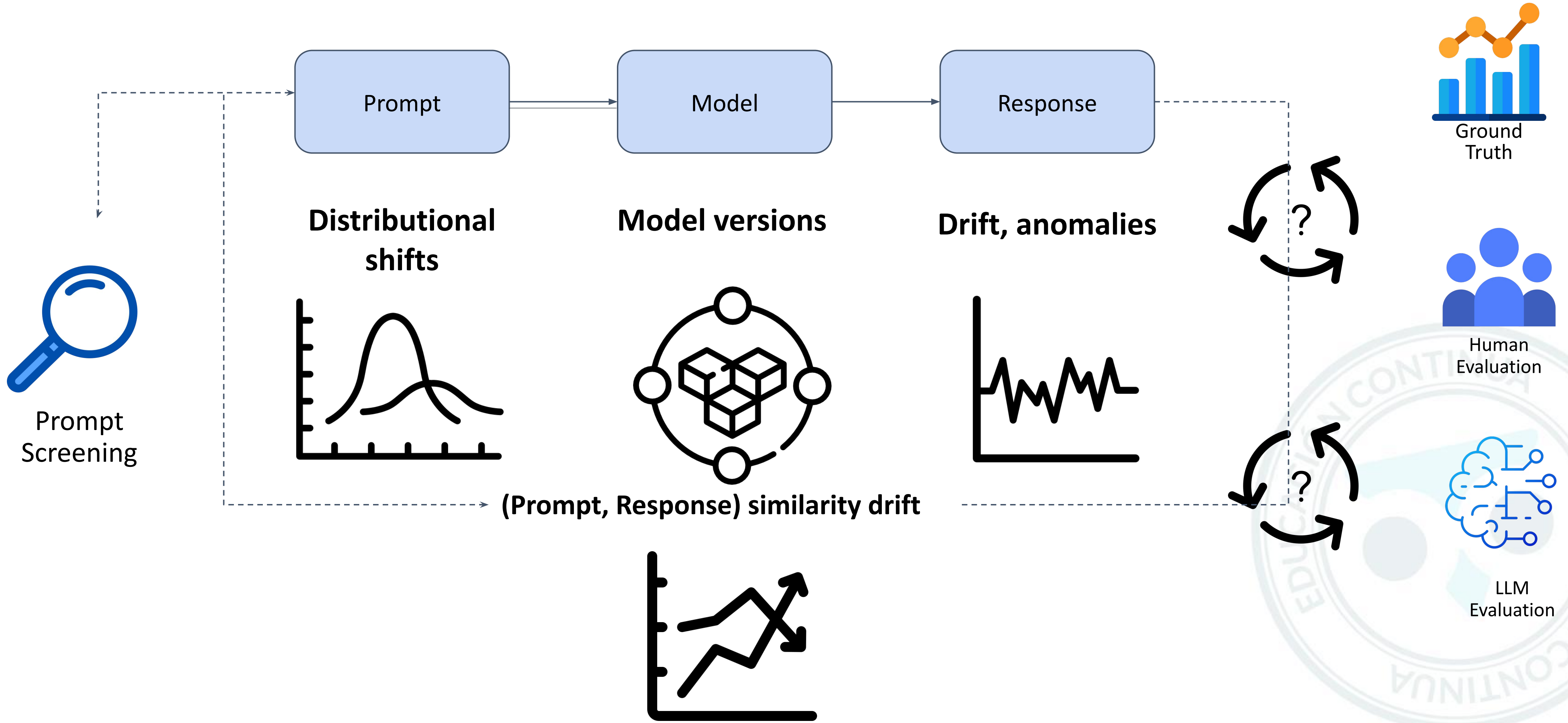
Monitoring and maintenance of LLM applications

MLOps

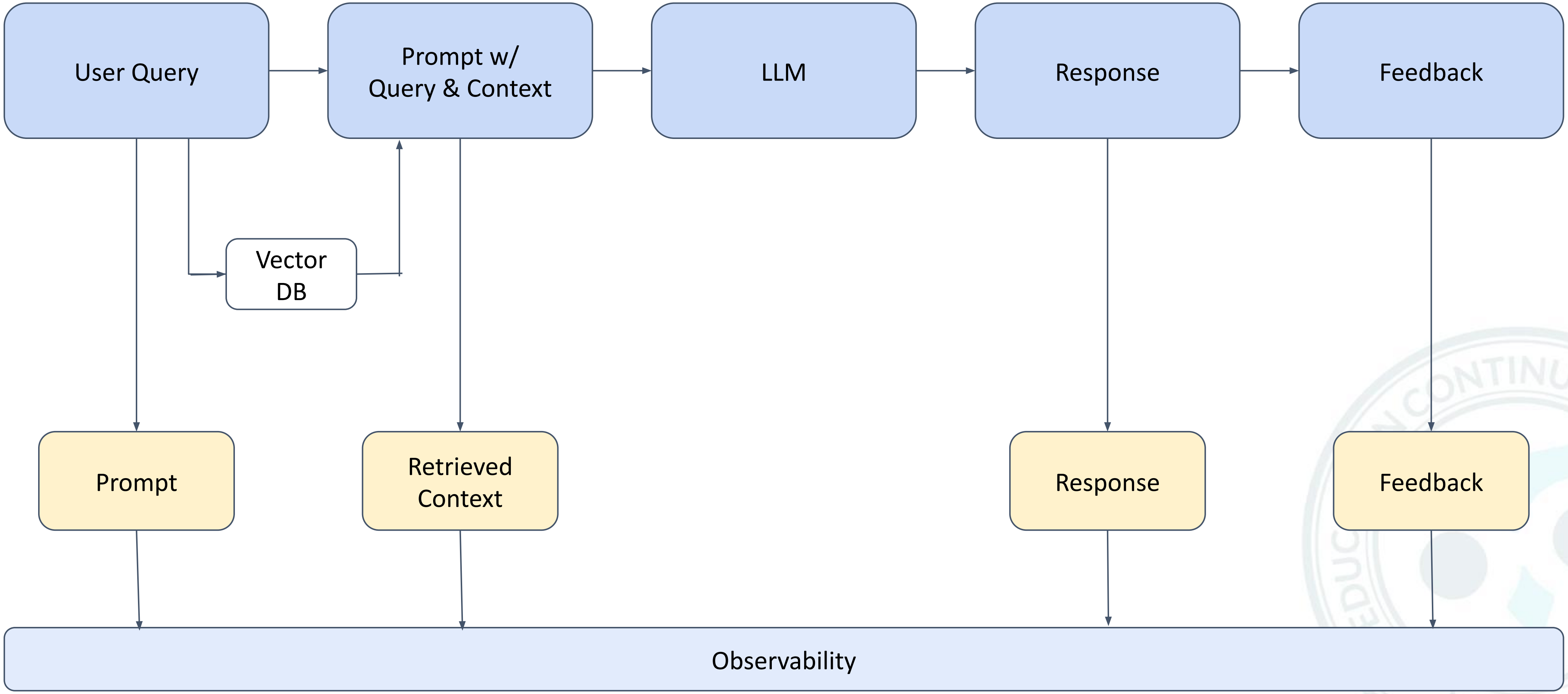
Monitoring for 'classic' ML



Monitoring for LLMs



MLOps


User Query
Prompt w/
Query & Context
LLM
Response
Feedback
Vector
DB
Prompt
Retrieved
Context
Response
Feedback
Observability

Deploying and monitoring systems

Key advice

Use flexible tooling for packaging

Why?

- You will swap AI libraries over time: LangChain, LLamaIndex, Python, …
- Uniform APIs lower the cost of switching libraries for a use case

How?

- MLflow supports built-in-flavors, PyFuncs, and custom flavors.
- All are managed behind uniform APIs

Use optimized inference

Why?

- User experience and TCO

How?

- Real-time: Model Serving
  - Foundation Model APIs for pre-optimized architectures
  - Custom models for DIY
- Batch and streaming
  - ai_query to call Model Serving
  - GPU clusters with vLLM, etc.

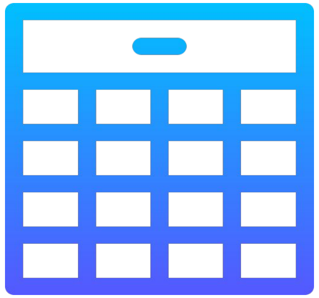
Performance evalaution and continous improvement of LLM applications

LLM evaluation methods & resources

Ground-truth metrics

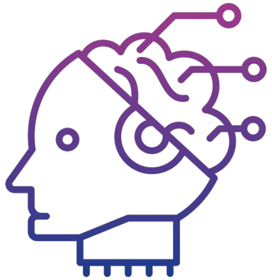
- BLEU
- ROGUE
- METEOR

Benchmarks



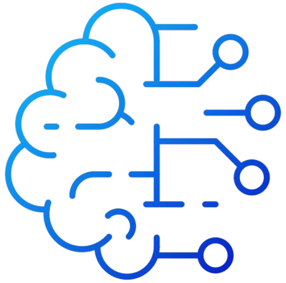
- TruthfulQA
- Arc
- HellaSwag

Uncertainty estimation



- SelfCheckGPT
- Perplexity

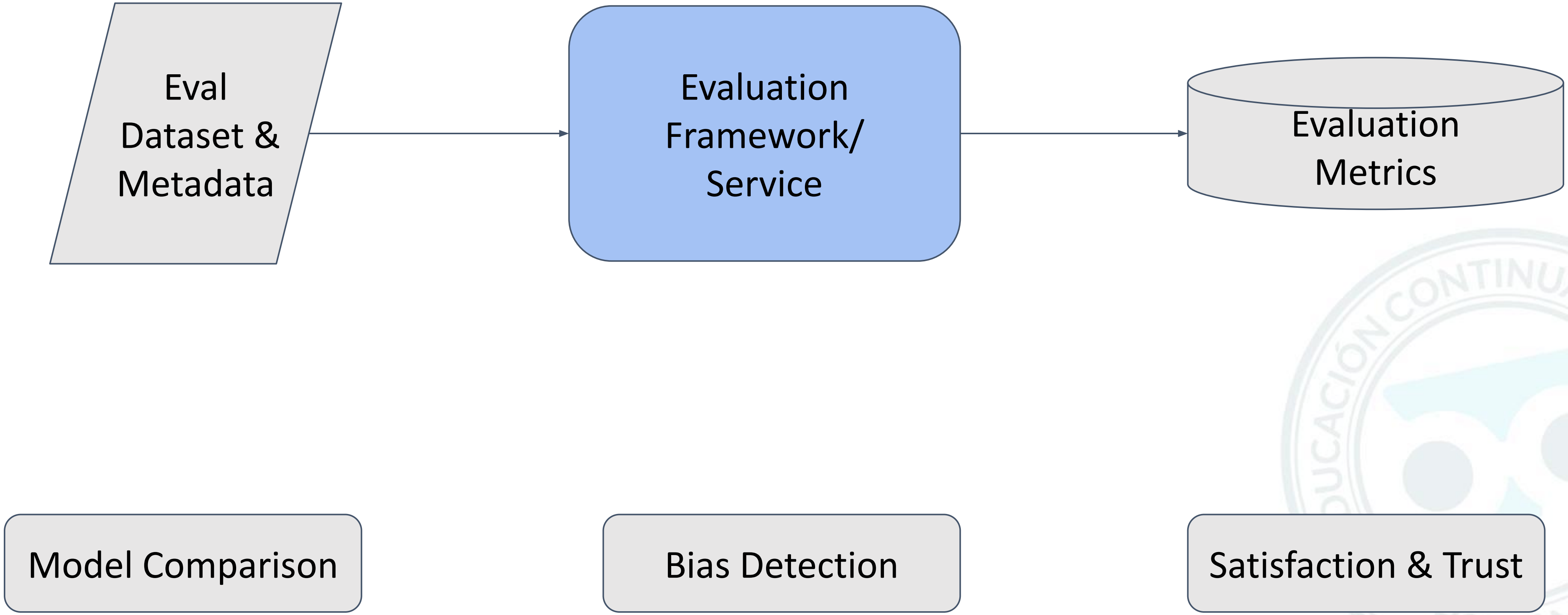
LLM evaluation of LLMs



- Prometheus
- JudgeLM

MLOps

Evaluation - One of the Most Impactful Parts



MLOps

Evaluating Gen AI systems

Key advice

Augment existing eval tooling

Why?

- Much tooling is reusable: MLflow, data pipelines, etc.
- New metrics can be added to existing systems

How?

- Adopt metrics from classic areas: toxicity (NLP), precision/recall (IR), …
- Use new tools like LLM-as-a-judge
- Evaluate both the components + system as a whole

Build user feedback into your app

Why?

- Users can be the best judges
- Build proprietary datasets for the future fine-tuning and pretraining

How?

- Consider implicit and explicit feedback
- Manage feedback like any other data: same governance, same ETL, etc.

MLOps

Evaluating Gen AI systems with mlflow

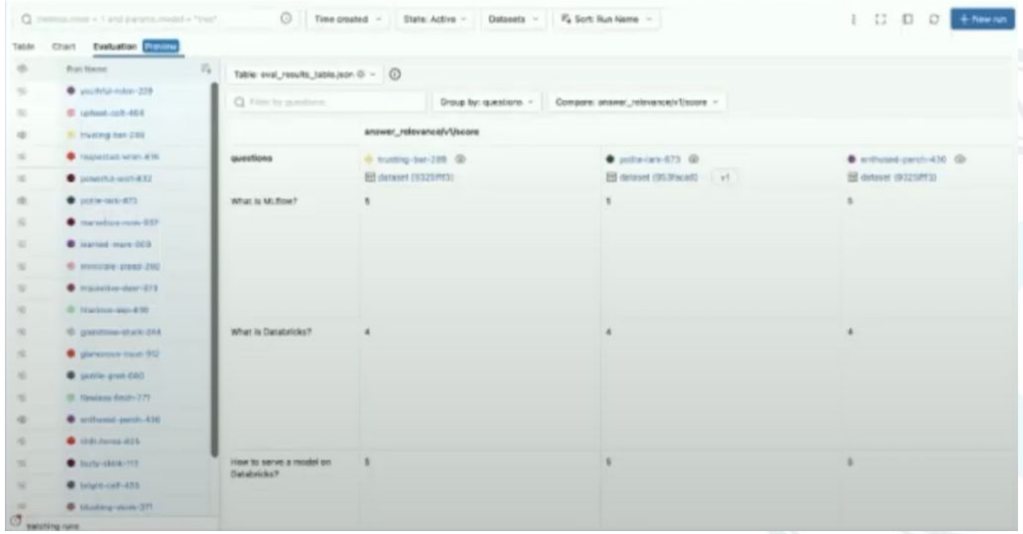
Batch evaluation in code

- LLM-as-a-judge
- Human evaluation using ground truth data
- New metrics for Gen AI, NLP, and retrieval

```python
from mlflow.metrics.genai.metric_definitions import answer_relevance

answer_relevance_metric = answer_relevance(model="endpoints:/gpt-4")

results = mlflow.evaluate(
  model,
  eval_df,
  model_type="question-answering",
  evaluators="default",
  predictions="result",
  extra_metrics=[answer_relevance_metric, mlflow.metrics.latency()],
  evaluator_config={
    "col_mapping": {
      "inputs": "questions",
      "context": "source_documents",
    }
  }
)
print(results.metrics)


results.tables["eval_results_table"]
```

Interactive evaluation in UI

- Compare Multiple models and prompts visually
- Iteratively test new queries during development



Materials

Class Repo https://github.com/LGuillermoAngaritaG/llmops-class-eia

Books: https://www.databricks.com/sites/default/files/2024-06/2023-10-EB-Big-Book-of-MLOps-2nd-Edition.pdf

List of applications for LLMOps: https://github.com/tensorchord/Awesome-LLMOps

Courses: https://github.com/mlabonne/llm-course

UNIVERSIDAD EIA

¡TU OPINIÓN CUENTA!

Estudiante de Educación Continua, te invitamos a completar nuestra breve encuesta de satisfacción.

¡Ayúdanos a mejorar tu experiencia educativa!