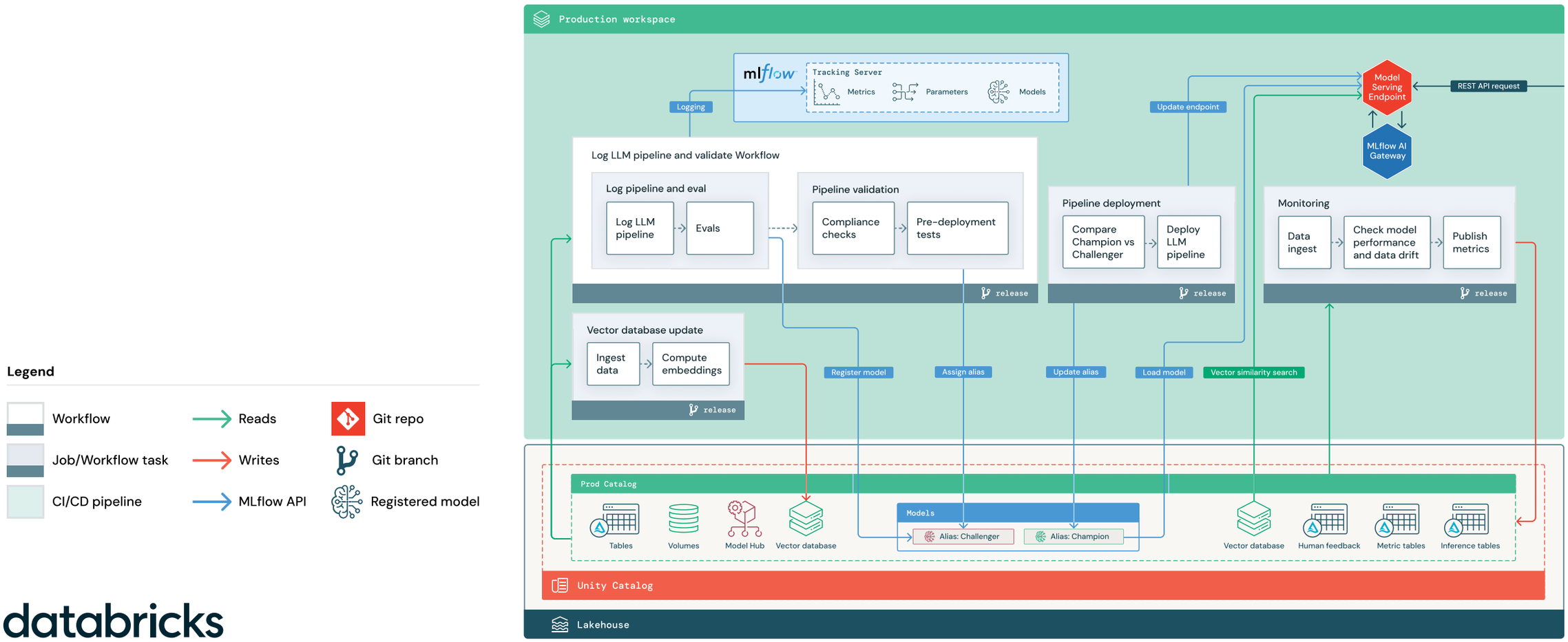
Reference architecture

To illustrate potential adjustments to your reference architecture from traditional MLOps, we provide a modified version of the previous production architecture for two separate LLM-based applications:

1. RAG workflow using a third-party API
2. RAG workflow using a self-hosted fine-tuned model.

Note that in either of these examples, the retrieval element using the vector database could be removed, and the LLM queried directly through the Model Serving endpoint.

RAG with a third-party LLM API

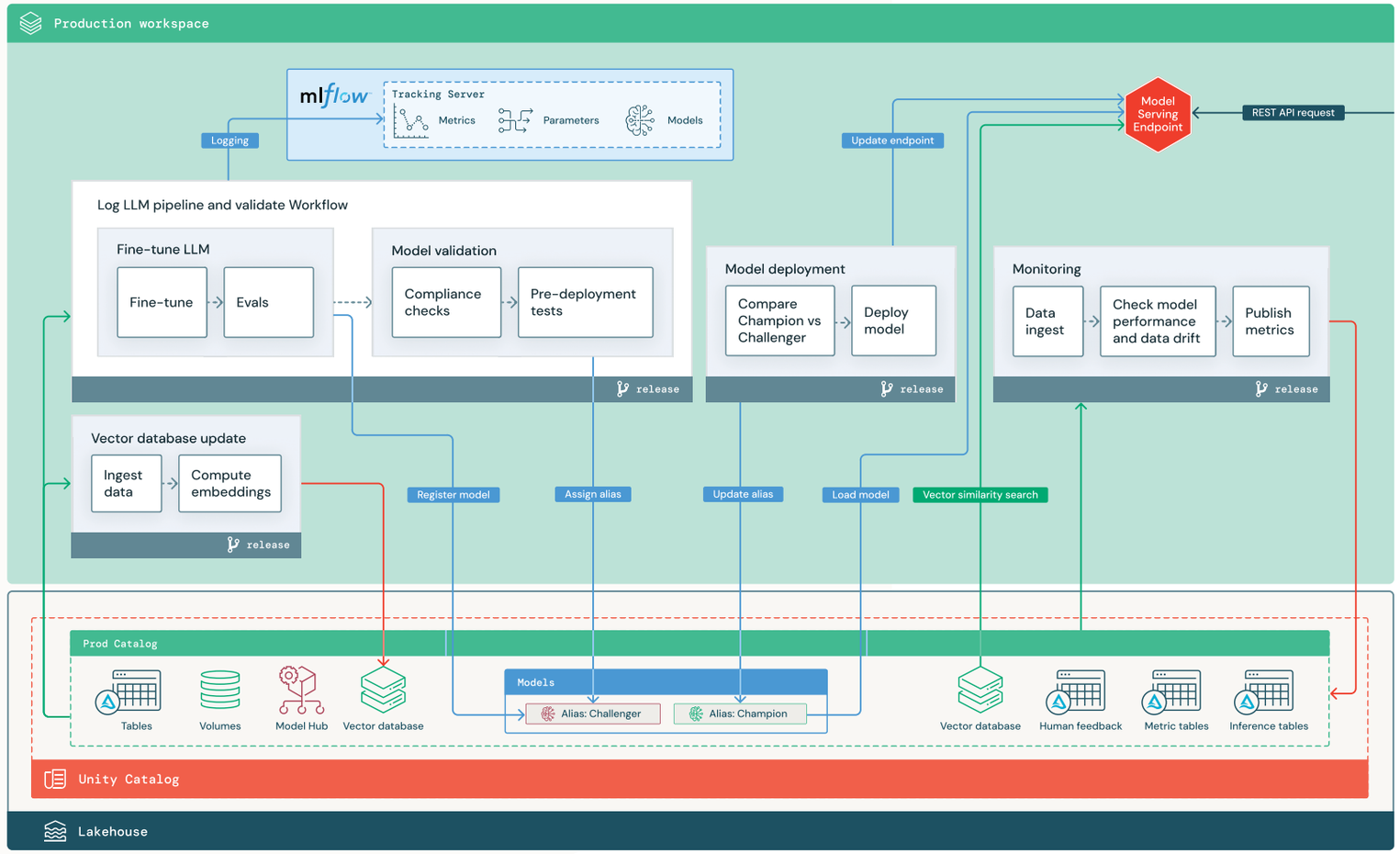


Legend

Workflow	Reads	Git repo
Job/Workflow task	Writes	Git branch
CI/CD pipeline	MLflow API	Registered model

databricks

RAG with a fine-tuned OSS model



Legend

- Workflow
- Job/Workflow task
- CI/CD pipeline
- Reads
- Writes
- MLflow API
- Git repo
- Git branch
- Registered model

databricks

The primary changes to the above production architectures are:

Model Hub

Since LLM applications often make use of existing, pretrained models, an internal or external model hub becomes a valuable part of the infrastructure. In the RAG with fine-tuned model example we illustrate using an existing base model from the model hub that is then fine-tuned in production.

Vector Database

Some (but not all) LLM applications use vector databases for fast similarity searches, most often to provide context or domain knowledge in LLM queries. To ensure that the deployed language model has access to up-to-date information, regular vector database updates can be scheduled as a Databricks job. Note that the logic to retrieve from the vector database and inject information into the LLM context can be packaged in the model artifact logged to MLflow using MLflow LangChain or PyFunc model flavors.

MLflow AI Gateway

In LLM-based applications where a third-party LLM API is used, the MLflow AI Gateway can be used as a standardized interface to route requests from vendors such as OpenAI and Anthropic. In addition to providing an enterprise-grade API gateway, the AI Gateway centralizes API key management and provides the ability to enforce cost controls.

Fine-tune LLM

Instead of a de novo model training pipeline, LLM applications will generally fine-tune an existing model (or use an existing model without any tuning). Fine-tuning is a lighter-weight process than training, but it is similar operationally. We represent model fine-tuning and model deployment as separate Databricks Workflows given that validating a fine-tuned model prior to deployment may be a manual process involving a human-in-the-loop.

Model Serving

In the case of RAG using a third-party API, one key architectural change is that the LLM pipeline will make external API calls, from the Model Serving endpoint to internal or third-party LLM APIs. It should be noted that this adds complexity, potential latency, and another layer of credential management. By contrast, in the fine-tuned model example, the model and its model environment will be deployed.

Human feedback in monitoring and evaluation

Human feedback loops may be used in traditional ML but become essential in most LLM applications. Human feedback should be managed like other data, ideally incorporated into monitoring based on near real-time streaming.

CHAPTER 7

Conclusion

In an era defined by data-driven decision making and intelligent automation, the importance of MLOps cannot be overstated. MLOps provides the essential scaffolding for developing, deploying, and maintaining AI models at scale, ensuring they remain accurate and continue to deliver business value. The emergence of LLMOps highlights the rapid advancement and specialized needs of the field of Generative AI. However, at its heart, LLMOps is still rooted in the foundational principles of MLOps.

Whether you are implementing traditional machine learning solutions or LLM-driven applications, the four core tenets remain constant:

- **Business goal:** Always keep your business goals in mind
- **Data-centric:** Prioritize a data-centric approach
- **Modular:** Implement solutions in a modular manner
- **Automated:** Aim for processes to guide automation

Databricks stands uniquely positioned as a unified, data-centric platform for both MLOps and LLMOps. Serving as the foundation, Unity Catalog provides a single governance solution for all data and AI assets. This is complemented by MLflow for experiment tracking, Model Serving for real-time deployment, Lakehouse Monitoring to ensure long term efficiency and performance stability, and Databricks Workflows to seamlessly orchestrate data pipelines.

As we look forward to the oncoming wave of AI advancements, it's clear that employing a robust MLOps strategy will remain central to unlocking AI's full potential. With firm MLOps foundations in place, organizations will be able to maximize their AI investments to drive innovation and deliver business value.

databricks