Exploring MLOps and LLMOps:

Architectures and Best Practices

Arpit Jasapara, Yinxi Zhang, Joseph Bradley
Databricks

About us

Arpit Jasapara

Engineering

Yinxi Zhang

Data Science

Joseph Bradley

Product Specialist

Agenda

MLOps + LLMOps

MLOps

- Talk (20 minutes)
- MLOps Stacks Demo (15 minutes)
- Q&A (10 minutes)

LLMOps

- Talk & MosaicAI Agent Framework Demo (35 minutes)
- Q&A (10 minutes)

MLOps

- What is MLOps?
  - Why should I care?
  - Best practices
- MLOps on Databricks
  - Unity Catalog
  - Model Serving
  - Monitoring
- Databricks **MLOps Stacks** Demo

What is MLOps?

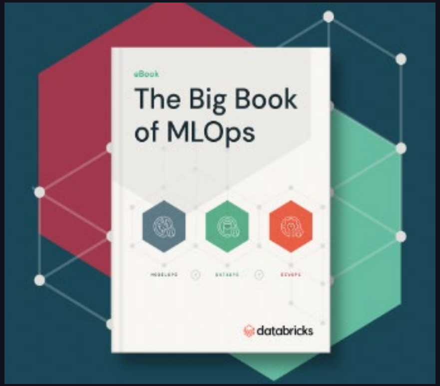
MLOps = DataOps + DevOps + ModelOps

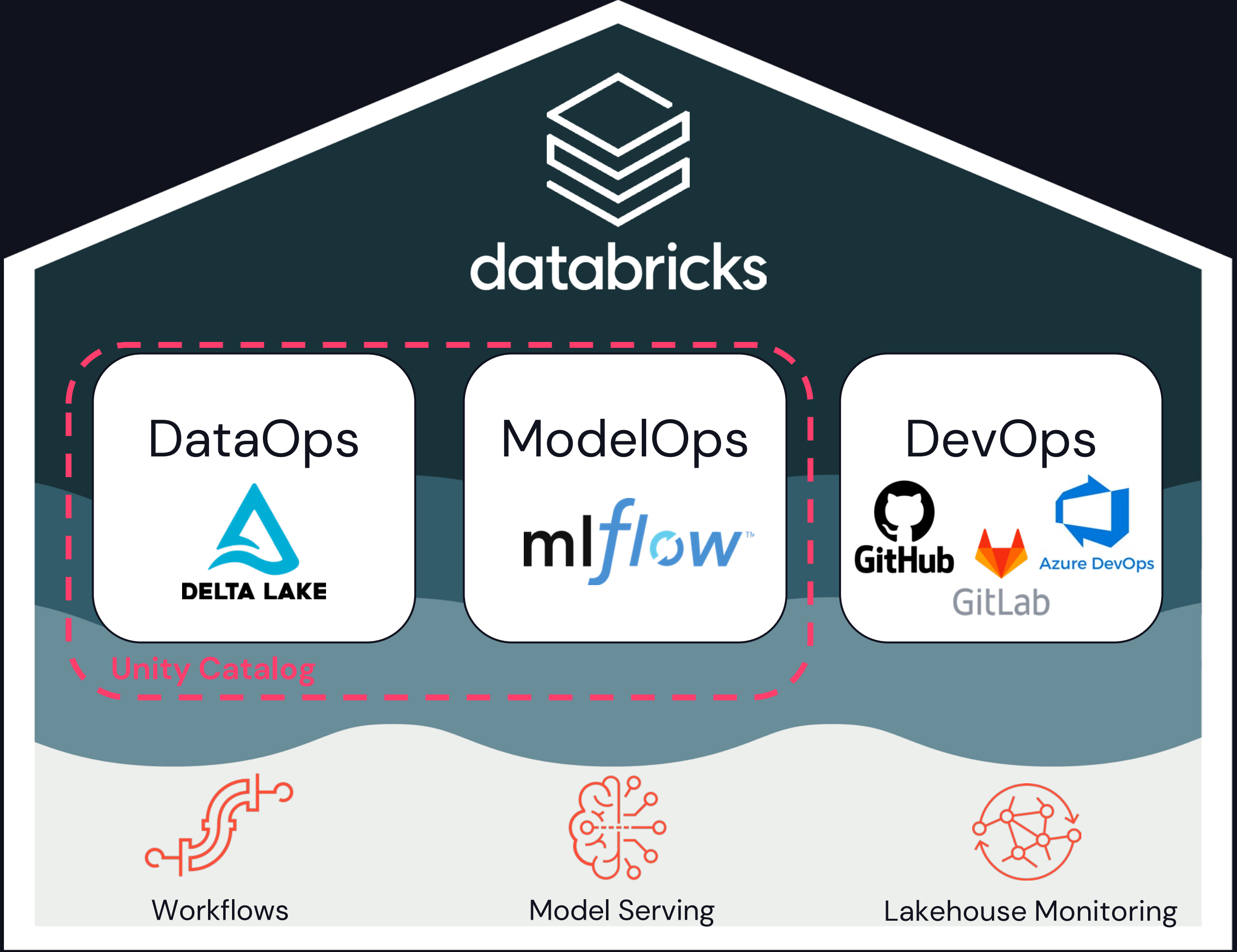
MLOps is the set of **processes and automation**

for **managing data, code, and models**

to improve **performance, stability, and efficiency** of ML systems

Big Book of MLOps




databricks
DataOps
DELTA LAKE
ModelOps
mlflow
DevOps
GitHub
GitLab
Azure DevOps
Unity Catalog
Workflows
Model Serving
Lakehouse Monitoring

Why should I care about MLOps?

MLOps helps you reduce risk

- Technical risk - poorly performing models, fragile infrastructure
- Compliance risk - violating regulatory or corporate policies

MLOps improves long-term efficiency through automation

- Streamline delivery of models to production
- Catch errors before they hit production
- Avoid slow, manual processes

Best Practices: Roles and Process

Business Stakeholder → Responsible for business value of the ML solution

Data Engineer → Builds data pipelines

Data Scientist (DS) → Translates business problem; trains, tunes model

ML Engineer (MLE) → Deploys ML model to production

Data Governance Officer → Responsible for data governance and compliance

Best Practices: Roles and Process

Best Practices: Environment Isolation

AI Assets:

Code
- Training
- Inference
- …

Data
- Features
- Labels
- …

Model
- MLflow Model (sklearn, …)

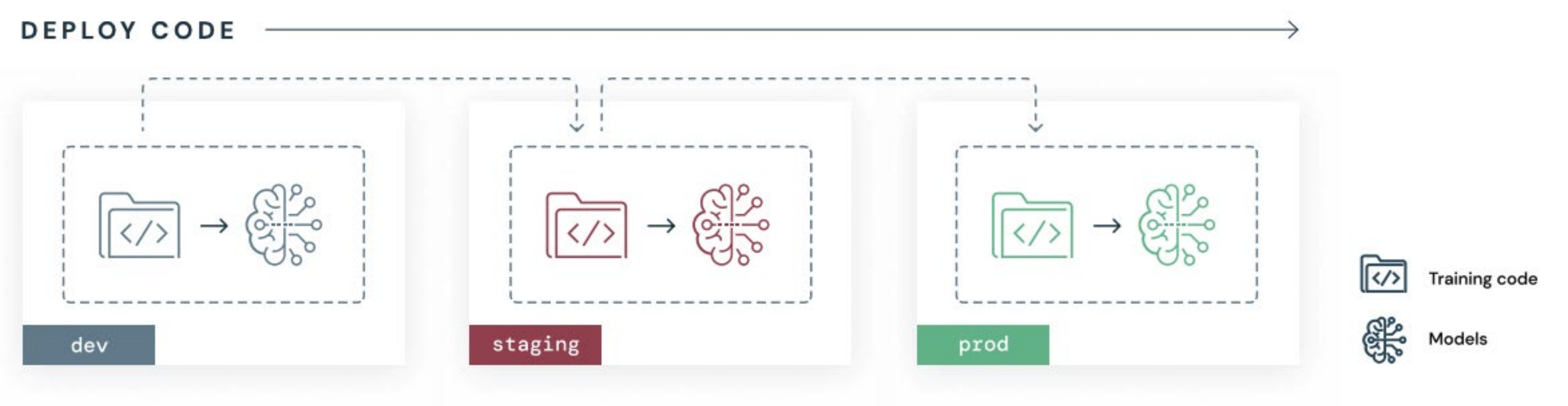
Assets need to be:

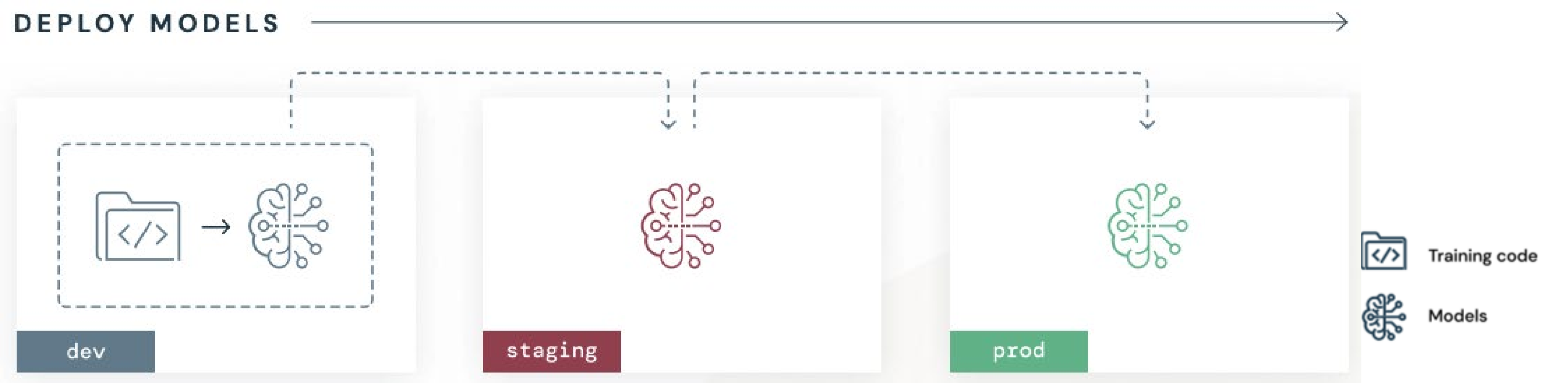
Developed	Tested	Deployed
`dev`	`staging`	`prod`

Trust, quality, and testing: **Low** ⟷ **High**

Openness of access: **Open** ⟷ **Locked-down**

Best Practices: Deployment Patterns





Best Practices: Deploy Code

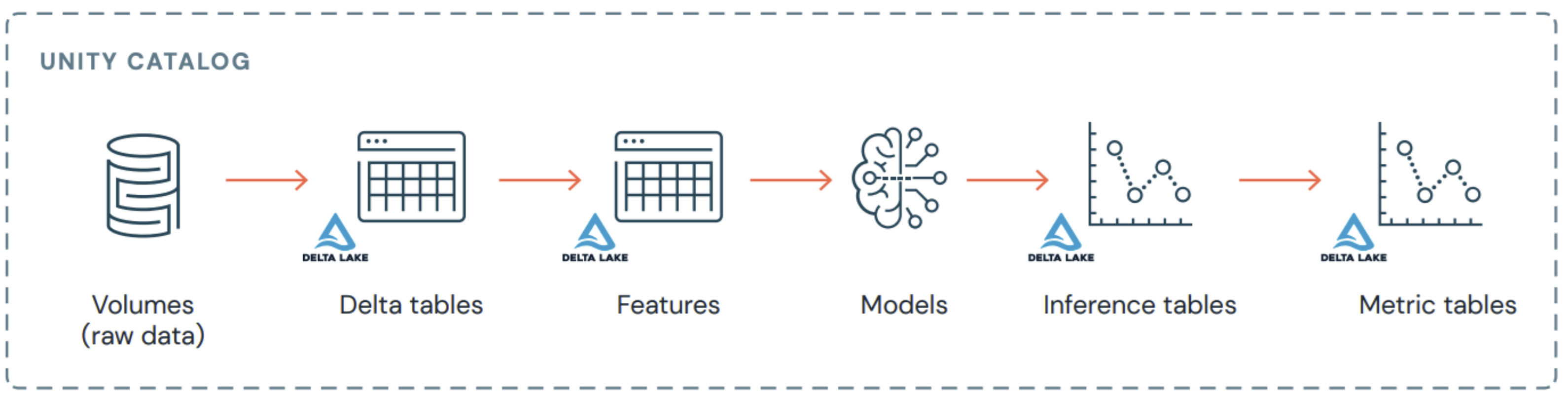
Process

Development	Staging	Production
⚙️ Develop training code	✓ Test model training code on subset of data	✓ Train model on production data
⚙️ Develop ancillary code	✓ Test ancillary code	✓ Test model
→ Promote code	→ Promote code	→ Deploy model and ancillary code

Benefits

Automation	Supports automated retraining in locked-down environments.
Data access control	Only production environment needs read access to production training data.
Reproducible models	Engineering control over training environment, which helps to simplify reproducibility.
Support for large projects	This pattern forces modular code and iterative testing, helping coordination and development.

MLOps on Databricks: Unity Catalog



Single governance solution for **data and AI assets**:

- Centralized access control
- Auditing
- Lineage (tables, features, models, workflows, etc.)
- Discovery
- Sharing assets between workspaces

MLOps on Databricks: Model Serving

Deploy models as a **real-time API** to integrate model predictions with applications or websites.

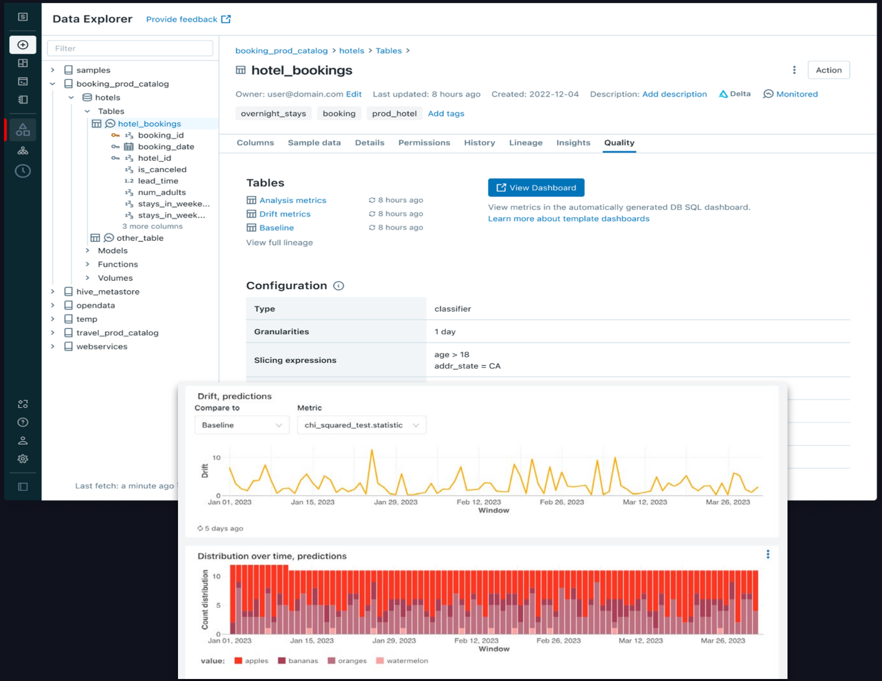
Databricks Integrated	Simplified, Serverless Deployment	Production Ready	Inference Tables
Automatic feature/vector lookups, monitoring, and unified governance	Deploy any model type on CPU or GPU with scalable, automated container build and very low latency (p50 <10ms) & high throughput (QPS >25k)	Supports online evaluation strategies such as A/B testing through the ability to serve multiple models to a serving endpoint	Log each request & response for monitoring, retraining, debugging, and more

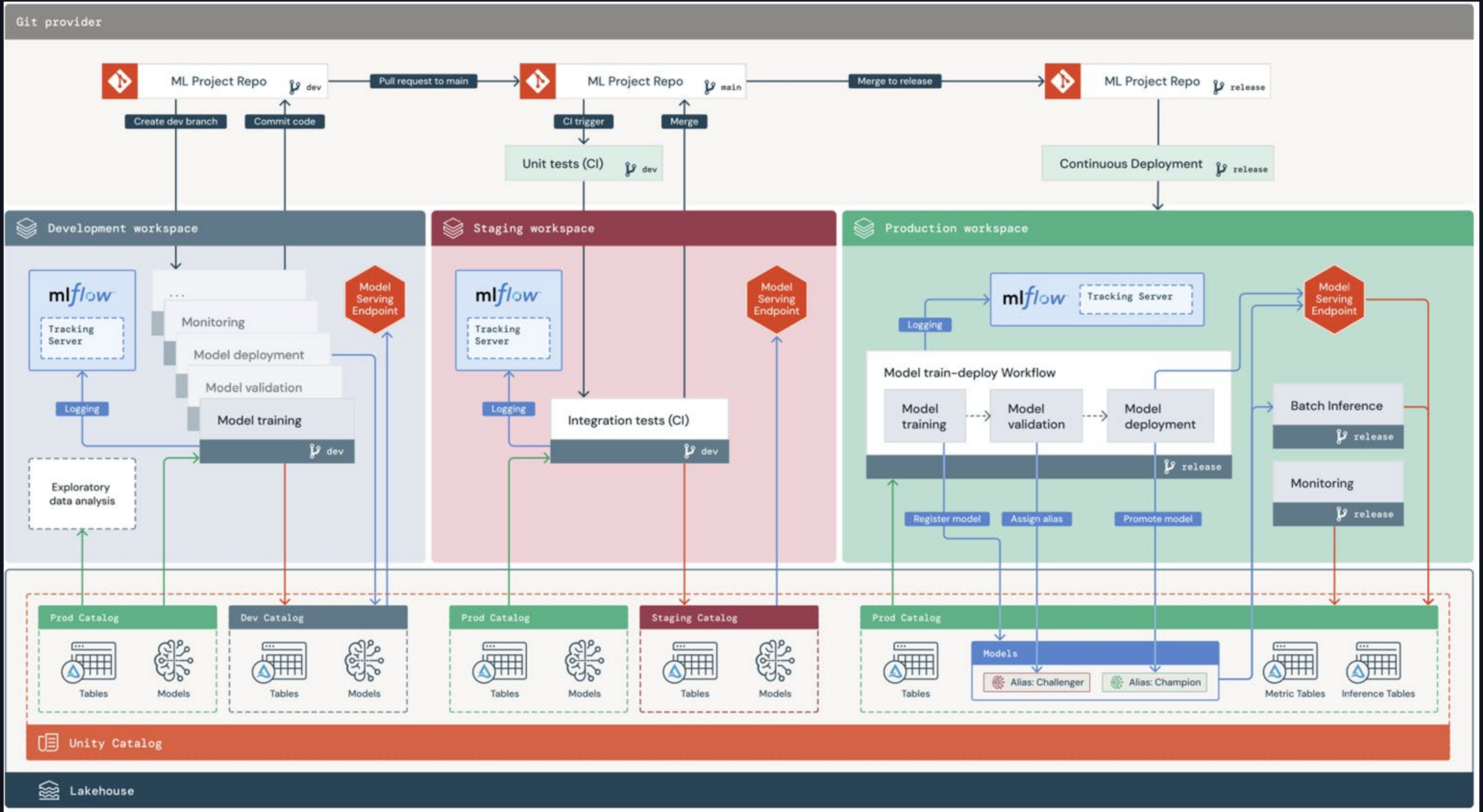
MLOps on Databricks: Monitoring

Data-centric monitoring solution to ensure that both data and AI assets are of high quality, accurate, and reliable.

- Incrementally processes data in UC tables
- Calculates **profile metrics** and **drift metrics**
- Supports **custom metrics** as SQL expressions
- **Auto-generates DBSQL dashboard** to visualize metrics over time

For MLOps, use in conjunction with **inference tables to monitor models**




Git provider
ML Project Repo
dev
Pull request to main
ML Project Repo
main
Merge to release
ML Project Repo
release
Create dev branch
Commit code
CI trigger
Merge
Unit tests (CI)
dev
Continuous Deployment
release
Development workspace
Staging workspace
Production workspace
mlflow
Tracking Server
...
Monitoring
Model deployment
Model validation
Model training
dev
Model Serving Endpoint
Logging
Exploratory data analysis
mlflow
Tracking Server
Logging
Integration tests (CI)
dev
Model Serving Endpoint
mlflow
Tracking Server
Logging
Model Serving Endpoint
Model train-deploy Workflow
Model training
Model validation
Model deployment
release
Batch Inference
release
Monitoring
release
Register model
Assign alias
Promote model
Prod Catalog
Tables
Models
Dev Catalog
Tables
Models
Prod Catalog
Tables
Models
Staging Catalog
Tables
Models
Prod Catalog
Tables
Models
Alias: Challenger
Alias: Champion
Metric Tables
Inference Tables
Unity Catalog
Lakehouse

Databricks MLOps Stacks

A customizable framework for managing the ML lifecycle on Databricks, following all **MLOps best practices**

- ➔ Create and manage **production MLOps infrastructure** on Databricks
- ➔ Integrates with common CI/CD providers like **GitHub** and **Azure DevOps**
- ➔ Manage AI assets (experiments, features, models, monitoring, etc.) and workflows as **IaC with Databricks Asset Bundles**

MLOps Stacks allow you to focus on **ML**, not infrastructure

- ➔ DS get started with **project development option** in Stacks
- ➔ MLEs easily set up CI/CD and customize the architecture as needed via the **CI/CD option** in Stacks
- ➔ DS then safely deploy project to production through secure CI/CD pipelines and workflows setup by the MLEs

Databricks MLOps Stacks Demo

MLOps Q&A

LLMOps

- What changes with Gen AI and LLMs?
- Key components for Gen AI
- Architecture and AI security
- Demo

Reminder: Shipping ML from Dev to Prod

ML Assets:

Execution Environment

Code
- Training
- Inference
- …

Data
- Features
- Labels
- …

Models
- MLflow Model (sklearn, …)

Assets need to be:

Developed	Tested	Deployed
dev	staging	prod

Shipping Gen AI from Dev to Prod

Gen AI Assets:

Execution Environment

Code
- Chain
- Inference
- …

Data
- Features
- Vector DB
- …

Models
- MLflow Model (LangChain, …)

Assets need to be:

Developed	Tested	Tested
dev	staging	prod

MLOps - What changes with Gen AI?

<table>
<tr><th>Properties of Gen AI models</th><th>Implications for MLOps</th></tr>
<tr><td>Models come in many forms:
• General vs. domain/task-specific models
• Proprietary vs. OSS
• Model-as-a-service APIs vs. self-managed
• Existing vs. custom fine-tuned vs. custom pretrained models</td><td rowspan="5">• Development process
• Legal concerns
• API governance
• Packaging artifacts
• Serving infrastructure
• Custom models
• Evaluation</td></tr>
<tr><td>Models range widely in size:
• Top general models have 100 billions - trillion parameters
• Top domain/task-specific models may have billions of parameters</td></tr>
<tr><td>Models take natural language prompts (or other unstructured data) as input.</td></tr>
<tr><td>Models can be given prompts with examples and/or context.</td></tr>
<tr><td>Models are hard to evaluate via traditional ML metrics since there is often no single “right” answer.</td></tr>
</table>

LLMOps: Key components for Gen AI

- Selecting models
- Leveraging your own data
- Evaluating Gen AI systems
- Deploying and monitoring systems

Selecting models

Key advice

Plan to use a variety of models

Why?

- Cost/performance trade-offs
- Task/domain-specific models
- Model improvements

How?

- Unified APIs and governance
- Toolchain supporting arbitrary models and providers

Plan to build custom models

Why?

- Cost/performance trade-offs
- Task/domain-specific models
- Building IP and competitive edges

How?

- Collect data and feedback now
- Choose models carefully

Selecting models

Unified API for all types of models.

Integration with services including Feature Serving, Vector Search, and Lakehouse Monitoring.

Governance via Unity Catalog for model objects and AI Gateway for APIs.

Databricks Model Serving

Custom Models

Deploy any model as a REST API with Serverless compute, managed via MLflow. CPU and GPU.

Foundation Models

Curated Foundation Models provided behind simple APIs. Pay-per-token and provisioned throughput, including for fine-tuned versions.

External Models

Govern external models and APIs.

For model guidance, see for example: Best-in-class open source generative AI models for free commercial use

Leveraging your own data

Prompt Engineering	RAG, Agents, and Tools	Fine-tuning	Pre-training
Craft prompts to guide GenAI behavior	Combine a GenAI model with custom, enterprise data and tooling	Adapt a pre-trained GenAI model to specific domains or tasks	Train a GenAI model from scratch

More control and customization, but more compute and complexity

Few-shot examples Evaluation data	Vector database Feature serving Function serving SQL database ...	Domain-specific data (millions of tokens) Task-specific examples (1000s)	General and domain-specific data (billions of tokens)

Leveraging your own data

Key advice

Unify data governance

Why?

- Data will grow: Raw data, context for RAG, inference logs, evaluation metrics, feedback, ...
- Data will be reused across use cases

How?

- Unified management of all types: raw files, tables, embeddings, feature serving, vector indexes, logs, metrics, ...
- Unified governance of data + AI assets

Plan towards customization

Why?

- Your data is your competitive edge.

How?

- Work on platforms supporting fine-tuning and pretraining
- Start simple, create baselines, iterate.
- Add customization based on:
  - Volume and quality of data
  - Compute & latency requirements
  - Your domain or application

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 future fine-tuning and pretraining

How?

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

Evaluating Gen AI systems with mlflow

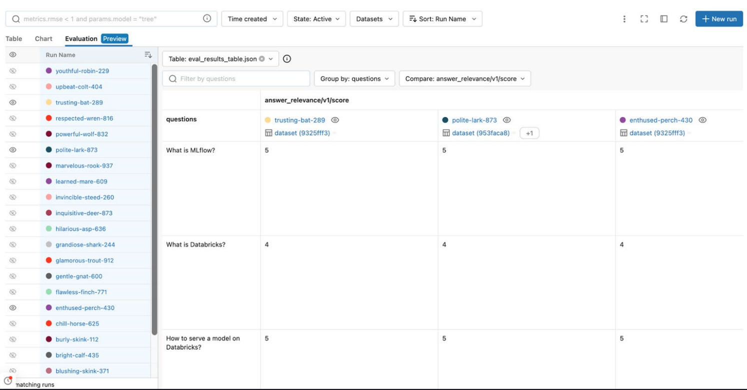
Batch evaluation in code

- 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



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.

Deploying and monitoring systems

Key advice

Your monitoring and core data/AI systems should be unified.

Why?

- Governance, lineage, and security are more important than ever with Gen AI.
- Inference logs, feedback, and metrics may be inputs for other AI systems.

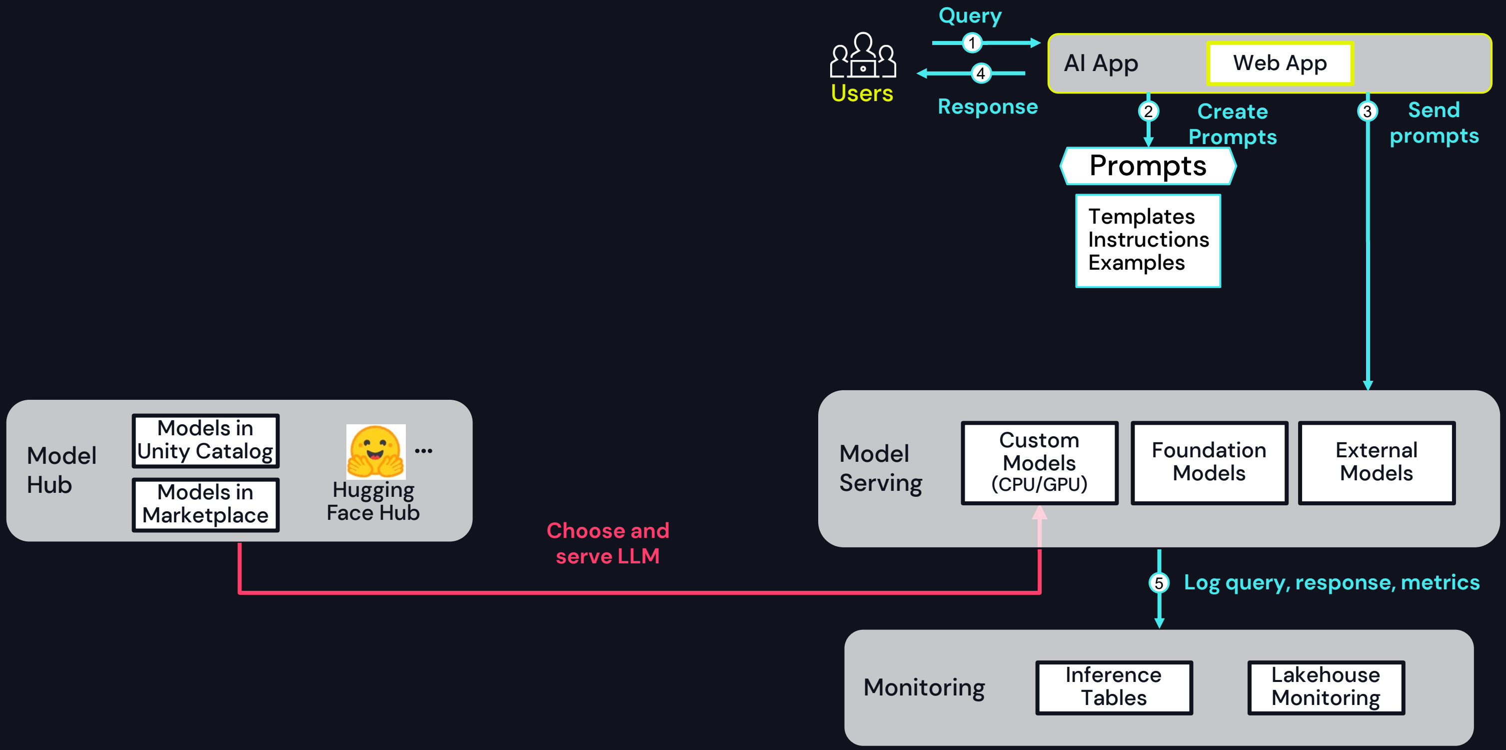
How?

- Unify governance, lineage, and access controls across data (inputs and outputs) and assets (data and AI) in your platform.
- Share data formats (such as Delta) efficiently usable by all systems.
- Share data pipeline tooling.

LLMOps: Architecture and AI security

- Reference architectures
  - Prompt engineering
  - RAG and agents
  - Fine-tuning
  - Pretraining
- Databricks AI Security Framework

Architecture: prompt engineering



Reusable infrastructure

Your initial GenAI use case will help you to assemble key pieces of your eventual GenAI + data platform.

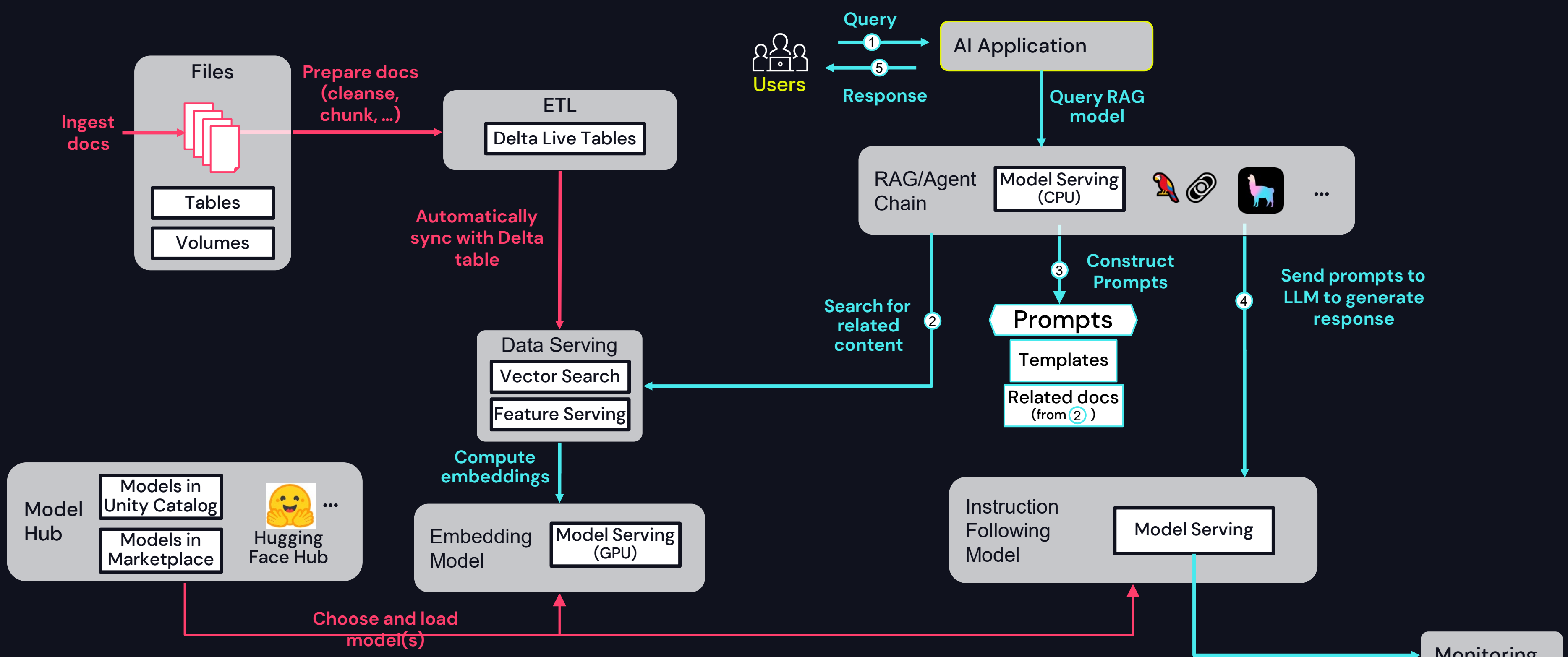
AI App
- Web App

Model Hub
- Models in Unity Catalog
- Models in Marketplace
- Hugging Face Hub
- ...

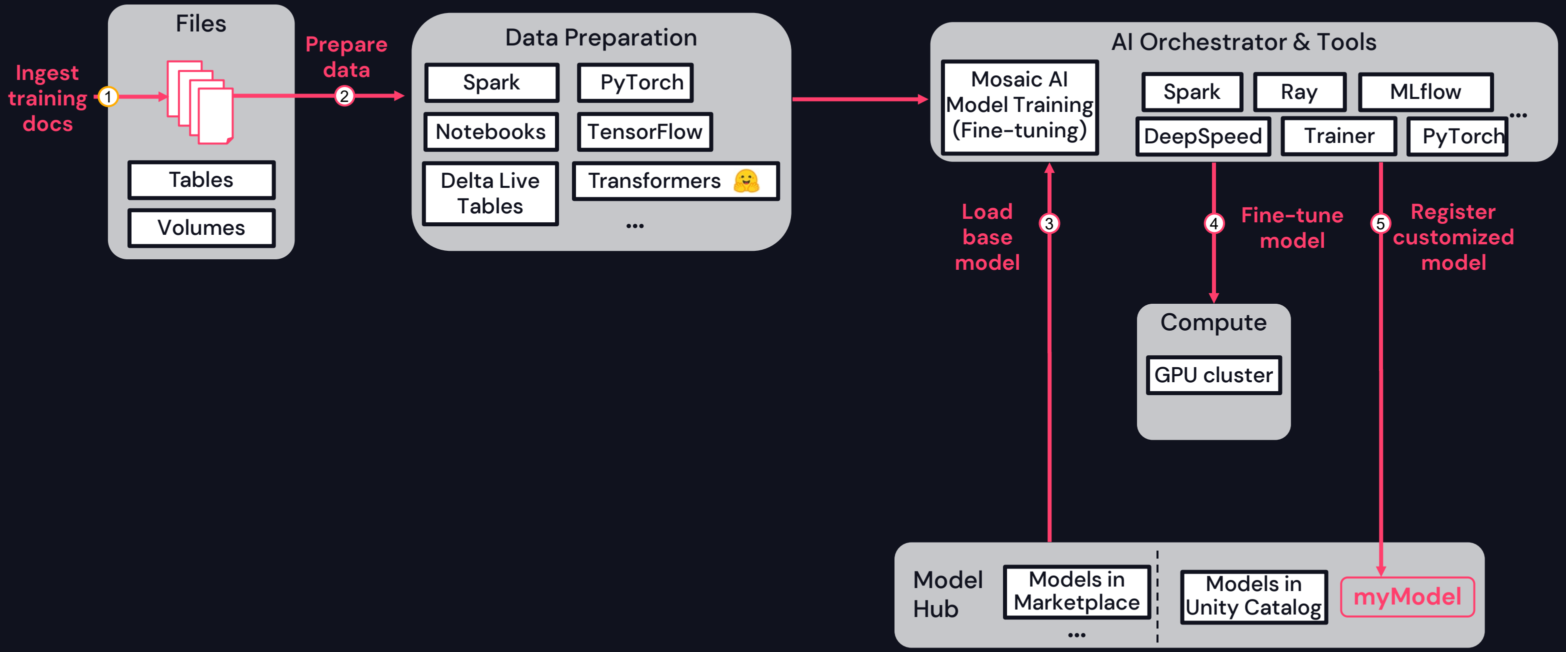
Model Serving
- Custom Models (CPU/GPU)
- Foundation Models
- External Models

Monitoring
- Inference Tables
- Lakehouse Monitoring

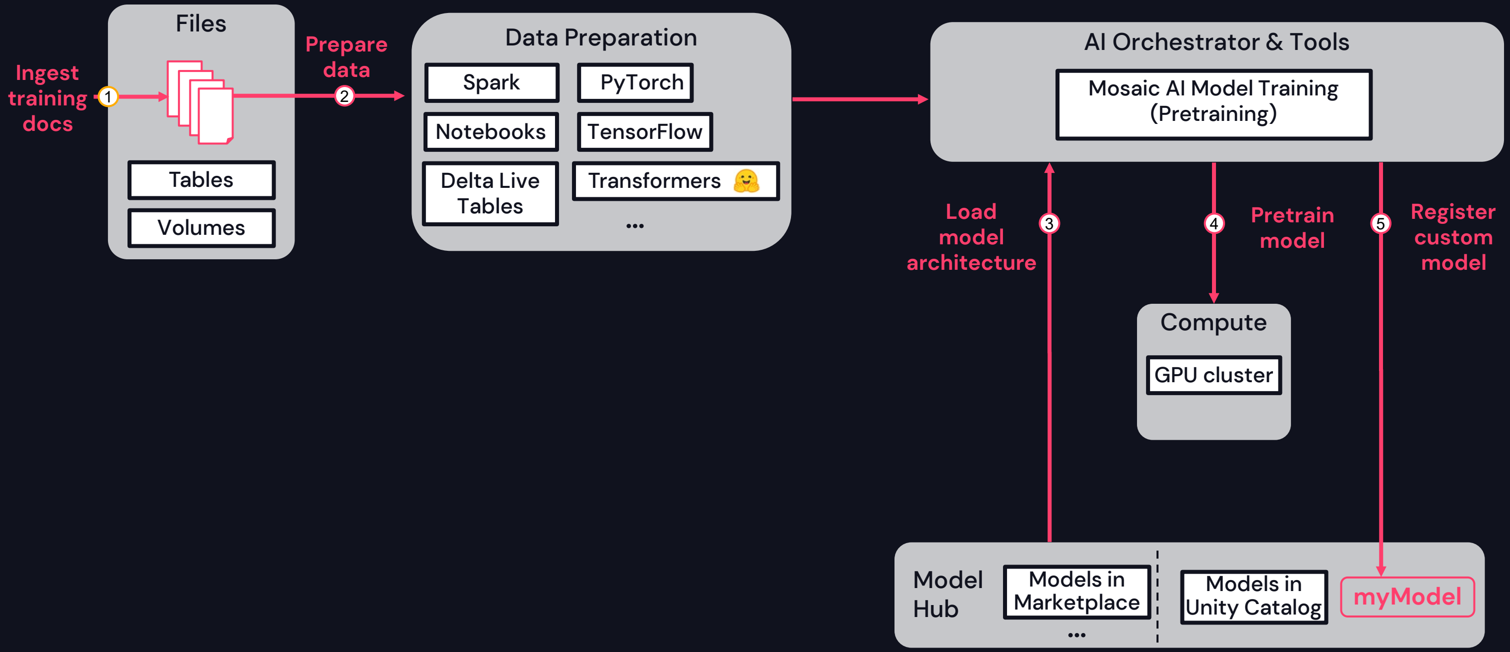
Architecture: RAG and agents



Architecture: fine-tuning



Architecture: pretraining



Pretraining a fully custom model

DBRX: Top-performing open-source, commercially viable LLM

DBRX

Designed for enterprise use

- Open-source for commercial use
- Base model can be fine-tuned
- Fast & accurate. For example, higher quality than Llama2-70B yet 2x faster for inference.

From March 2024

- Foundation Model APIs
- AI Playground
- Databricks Marketplace
- Hugging Face Hub & GitHub

In terms of operations, what did it take?

Pretraining a fully custom model

DBRX

It took Mosaic AI.

Composer for optimized deep learning training

Streaming Dataset for efficient data loading during training

LLM Foundry for training, fine-tuning and evaluating

Evaluation Gauntlet for evaluating quality

Notebooks and **Apache Spark** for data cleaning and processing

Delta Lake and **Unity Catalog** for data storage and governance

Mosaic Multi-Cloud Training (MCT) to train the model

MLflow and **Lakeview** for experiment tracking

Foundation Model APIs and **AI Playground** for eval and red-teaming

Databricks AI Security Framework

Holistic approach to AI system security

Recommendations on how to manage and deploy AI models safely and securely, by defining:

- 12 AI system components
- 55 technical AI risks
- 53 mitigating controls

Built with industry luminaries, partners, and customers.

Whitepaper

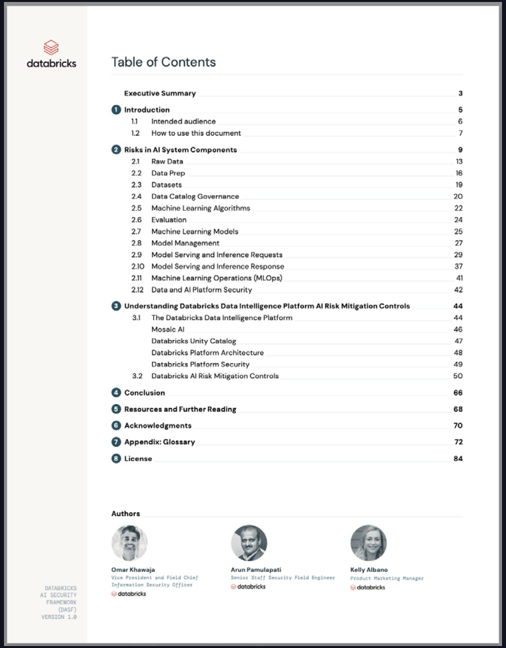
Databricks Security & Trust Center

databricks

Whitepaper

Databricks AI Security Framework (DASF)

Version 1.0



databricks

Table of Contents

Authors

Omar Khawaja
Vice President and Field Chief Information Security Officer
databricks

Arun Pamulapati
Senior Staff Security Field Engineer
databricks

Kelly Albano
Product Marketing Manager
databricks

DATABRICKS AI SECURITY FRAMEWORK (DASF) VERSION 1.0

GenAI at DAIS

Click here to access all GenAI sessions

Product Led Sessions

Top Databricks product sessions :

Mosaic AI Agent Framework / Quality Lab
Weds 11:20 AM - 12:00 PM | South, Level 2, Rm 211

Mosaic AI Vector Search
Tuesday 9:00 AM - 9:40 AM | South, Level 2, Rm 209

Mosaic AI Model Training
Thursday 12:30 PM - 1:10 PM | South, Level 2, Rm 211

Mosaic AI Deep Dive + Tools Catalog
Weds 12:30 PM - 1:10 PM | West, Level 2, Rm 2001

Shutterstock ImageAI, Powered by Databricks
Weds 11:20 AM - 12:00 PM | West, Level 2, Rm 2009

Top Customer led sessions
50 sessions. Recommended:

- JPMorgan
- Northwestern Mutual
- Corning
- Rolls-Royce
- CVS Health
- Fox
- Dun & Bradstreet
- Comcast

GenAI at DAIS

General Recommendations

Click here to access all GenAI sessions

Top Databricks led sessions
25 sessions. Recommended:

Beginner:

- Introduction To Mosaic AI: How Databricks Simplifies Your Genai Journey
- Introduction To Retrieval Augmented Generation And Implementing With Databricks
- Introduction To Vector Search On Databricks

Advanced:

- Deep Dive Into Building Production Quality Gen AI Applications
- Customizing Your Models: Rag, Fine-Tuning, And Pre-Training
- Deep Dive Into Mosaic AI : Getting Genai Apps To Production On Databricks
- How To Train Or Fine-Tune A Custom Llm On Your Data With Databricks

Top Customer led sessions
50 sessions. Recommended:

- JPMorgan
- Northwestern Mutual
- Corning
- Rolls-Royce
- CVS Health
- Fox
- Dun & Bradstreet
- Comcast

LLMOps Q&A

Learn more

- Read the Big Book of MLOps for more fundamentals and architecture
- Try out MLOps Stacks via the GitHub repo
- Try the RAG demo from the Databricks Demo Center