M2 release notes & upcoming releases

M2 TLDR

Overall

1. **MLflow Integration:** Most `rag_studio` SDKs are now part of `mlflow`: logging, evaluation, chain parameterization. MLflow Tracing integration is improved.
2. **Developer workflow improvements:** Simplified chain configuration that accepts Dictionaries and YAML.

Evaluation & Databricks LLM Judges

3. **Evaluation is faster & easier to use:** Faster evaluation with progress bar. Accepts single Pandas Dataframe as input and provides single output Dataframe with results. Metrics & data logged to MLflow vs. Delta Tables.
4. **Improved LLM Judges & Metrics docs:** Clarity in metric names and how metrics evaluate chains
5. **New metrics:** Added new metrics such as token count and latency
6. **LLM Judge Tuning:** Few-shot examples can enhance LLM Judge agreement with human ratings
7. **Customer-defined LLM Judges:** Custom judges can be defined to assess use-case specific criteria.

Logging & monitoring

8. **Human Feedback and Monitoring:** New Delta Tables that automatically ETL feedback collected via Review App and Feedback API

Chain compatibility

9. **Chains support External Models & Provisioned Throughput:** Chains can use any Model Serving model, including OpenAI and Provisioned Throughput models with automatic credential provisioning.
10. **Chains support Third-Party APIs:** Chains can utilize third-party APIs with secure credential management through Databricks Secrets.

Review App

11. **Review App SSO:** Review App is accessible to any SSO users even if they aren't a Databricks user

Data Pipelines

12. **Flexible data pipeline:** New data processing pipeline template supports easily testing out different parsing, chunking, embedding strategies - several off the shelf implementations are included.

Details and upcoming releases

Theme	M2 release	Upcoming releases
Integration with MLflow	RAG Studio's integration with MLflow is significantly improved for both chain developing / logging and chain evaluation. Evaluation: • `rag_eval.evaluate(...)` is now called via `mlflow.evaluate(..., model_type="databricks-rag")` • Evaluation metrics & evaluation data are logged to MLflow Runs (versus a Delta Table)	Customer defined LLM judges configuration will be moved from YAML into the native MLflow SDK

	Chain development: • `rag.log_model(...)` is merged into `mlflow.langchain.log_model(...)` SDK • Chain configuration via YAML is now part of MLflow e.g., `rag.RagConfig(...)` moved to `mlflow.models.ModelConfig(...)`	
One-click RAG Data Pipeline accelerator	Comes with various pre-built parsing / chunking / embedding strategies or build your own. 100% of the code is available to to you, so this accelerator can be fully customized	Improvements based on your feedback
Faster and easier to use evaluation capabilities	Beyond tighter integration with MLflow, Evaluation now: • Evaluation runs faster • Evaluation status displayed via a progress bar w/ ETA • **Breaking change** Improved schema for evaluation input and output • Accepts a single Pandas Dataframe as input vs. requiring 2 separate Delta Tables • Evaluation results are available as a single output table vs. split across assessments & metrics, making review of evaluation easier ○ Returned as Pandas Dataframes vs. saved to Delta Tables • Evaluation Set schema for retrieval ground truth is more flexible and does not require chunk text • **Breaking change** the config YML syntax has changed to accommodate the new functionality of custom judges and few-shot examples Documentation	Native MLflow-SDK approach for defining custom LLM judges and metrics.
Improved integration with MLflow Tracing	MLflow Tracing provides a way to instrument and visualize the steps in your chain. In this release, MLflow Tracing is more tightly integrated with RAG Studio: • When running your chain during local development, traces are logged to MLflow and visualized in a Notebook-based UI • The data from the trace can be accessed via the MLflow SDK	**Breaking change** Step-by-step chain traces are stored in the same MLflow tracing format used during development. This replaces the current trace schema. Traces collected online can be viewed using the same Mlflow UI as in development.

Support for External Models & Provisioned Throughput	Your chain can now use any External Model e.g., OpenAI, etc and any Provisioned Throughput model. Credentials for these models are automatically provisioned when calling `deploy_model()`. Previously, only FM API Pay-Per-Token models were available.	Support
Support for 3rd party APIs	Your chain can now use 3rd party API services e.g., non-Databricks vector databases, etc with secrets stored in Databricks Secrets Manager via environment variables. Note: Doing so requires manually updating the Model Serving endpoint after calling `deploy_model()`.	Native support for using Databricks Secrets via `deploy_model()`.
Fast chain deployment		If using a Databricks defined set of Python packages in your chain, chains now deploy to REST APIs and Review App in 2 - 3 minutes vs. 15 - 20 minutes.
Chain token streaming support		If your LangChain chain supports streaming, your deployed REST API and Review App will also support streaming. Support for token streaming in PyFunc based chains

Human feedback collection & online chain monitoring	Chat requests to a deployed chain's REST API and chat requests via the Review App are available as Delta Tables.
	
	These tables are ETL'd versions of the raw Inference Table that contains the same data, but has a raw schema and duplicate rows.
	
	2 tables are generated:
	• `request_log`: Keyed to request_id, contains one row for every trace. A trace represents one turn of conversation in a chat and is linked to other traces in that conversation via converastion_id. Contains:
	  ○ Request
	  ○ Response
	  ○ Step-by-step chain trace
	• `assessment_log`: Keyed to request_id, contains one row for every piece of human feedback collected via the Review App or the Feedback API, including
		Breaking change Improvements to the assessment log schema to enable the data to be more easily used as input to tune the LLM Judge models and create Evaluation Sets.

	feedback for live chats & offline review of traces. Contains: ○ Thumbs up / down + why ■ Free text & select-many choices ○ Corrected bot responses *Note: Due to how the ETL job is scheduled, rows in these tables can take up to ~2 hours to appear. Raw logs appear in the Inference Table in ~15 minutes.* Documentation	
Review App	Access to the Review App can be granted to any user in your company's SSO even if they do not have access to your Databricks Workspace. To use this feature, your account admin must enable SCIM. Documentation	Support for collecting 👍👎feedback on individual chunks, which can be used to generate Evaluation Sets and tune the Chunk Relevance LLM judge. Improvements to the UX of Review App that make reviewing simpler / faster for stakeholders.
LLM Judges & Metrics documentation	Improved the clarity of the metric & LLM Judge names. Added more clear documentation on what each metric & LLM judge evaluates and the data used. Documentation	How to guide & sample code explaining techniques to try based on the results of your evaluation.
New Metrics	Added token count and latency metrics.	Fully custom metrics e.g., use a user-defined function as a metric
Increasing the quality of Databricks provided LLM Judges	Improved support for providing few-shot examples to the LLM Judges to tune their agreement with human raters. `evaluate()` now accepts a Dataframe of examples that follows the same schema as the judges output, making it easier to modify incorrect judge scores / tag high quality judge scores and provide them as examples to improve the judge quality. Documentation	Improvements in the quality of the LLM Judges e.g., LLM judges are more likely to agree with human rated assessments.
Customer defined LLM judges	Added support for customer defined LLM judges. Each judge is defined by: • Aninstruction (e.g., "asses if the response follows my company tone of voice, which is xyz") • A set of trace fields for the judge to review (e.g.,	Customer defined LLM judges configuration will be moved from YAML into the native MLflow SDK

	request, response, ground truth, etc) • Optionally, 1+ examples of "good" and "bad" responses to tune the judge's quality Documentation	
Improvements to the chain developer workflow	Various improvements: • Chain configuration can be passed a Dictionary vs. only as a YAML file • Single wheel that is available on PyPi	`dbutils` use in the `chain.py` is automatically removed vs. the user needing to comment this code out before logging the model `delete_deployment()` API to turn off a Chain deployment
Improved support for other types of Chains	Support for custom PIP dependencies with a LangChain based chain	Support for logging PyFunc chains: • Rather than requiring PyFunc chains to be pickled/serialized (which often fails) - PyFunc-based Chains are captured as Code + Config • Support for non-Dataframe based PyFunc signature e.g., `predict(param1, param2)` vs. `predict(model_input : dataframe)`. PyFunc chains support YAML/Dictionary based configuration. **Breaking change** More flexible schema support for chains - you can now use custom parameters in your chain's input / output signature. Previously, you could only use one pre-defined signature for your chain.
Improved compatibility w/ security postures		Support for PrivateLink customers to use Model Serving with Vector Search securely Support for Azure Storage Firewall customers to use Inference Tables