2020 EDITION | UPDATED

Standardizing the Machine Learning Lifecycle

From experimentation to production with MLflow

Contents

Preface

Technology changes quickly. Data science and machine learning (ML) are moving even faster. In the short time since we first published this eBook, businesses across industries have rapidly matured their machine learning operations (MLOps) — implementing ML applications and moving their first models into production. This has turned ML models into corporate assets that need to be managed across the lifecycle.

That's why MLflow, an open-source platform developed by Databricks, has emerged as a leader in automating the end-to-end ML lifecycle. With 1.8 million[1] downloads a month — and growing support in the developer community — this open-source platform is simplifying the complex process of standardizing and productionizing MLOps. This updated eBook explores the advantages of MLflow and introduces you to the newest component: MLflow Model Registry. You'll also discover how MLflow fits into the Databricks Unified Data Analytics Platform for data engineering, science and analytics.

1 As of March 2020

CHAPTER 1: Machine Learning Lifecycle Challenges

Building machine learning models is hard. Putting them into production is harder. Enabling others — data scientists, engineers or even yourself — to reproduce your pipeline and results is equally challenging. How many times have you or your peers had to discard previous work because it was either not documented properly or too difficult to replicate?

Getting models up to speed in the first place is significant enough that it can be easy to overlook long-term management. What does this involve in practice? In essence, we have to compare the results of different versions of ML models along with corresponding artifacts — code, dependencies, visualizations, intermediate data and more — to track what's running where, and to redeploy and roll back updated models as needed. Each of these requires its own specific tools, and it's these changes that make the ML lifecycle so challenging compared with traditional software development lifecycle (SDLC) management.

This represents a serious shift and creates challenges compared with a more traditional software development lifecycle for the following reasons:

The diversity and number of ML tools involved, coupled with a lack of standardization across ML libraries and frameworks

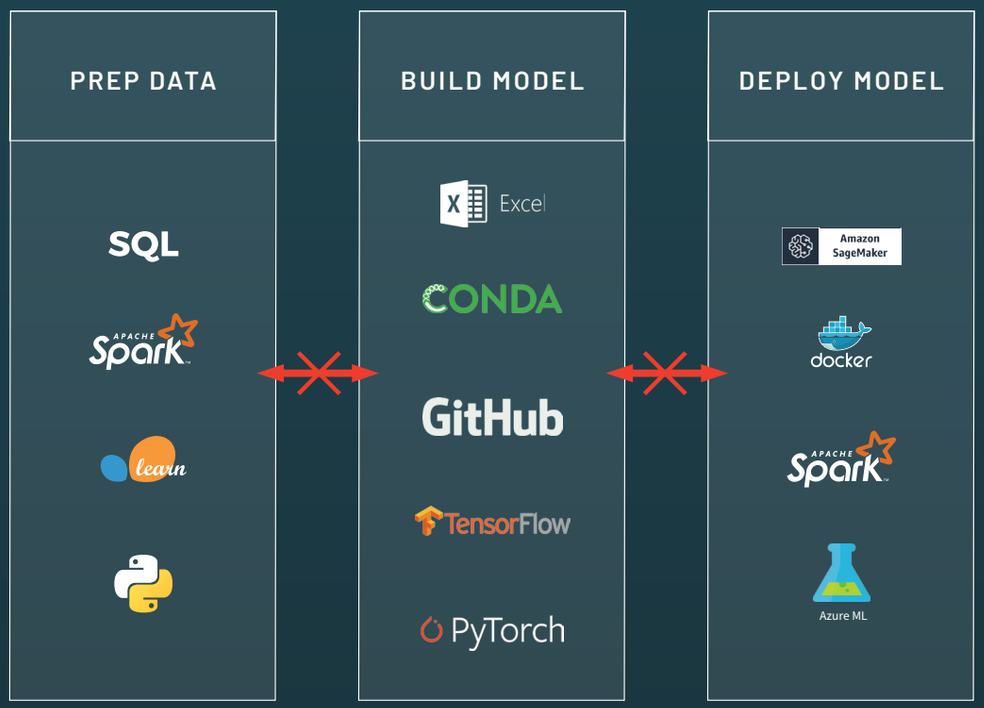
The continuous nature of ML development, accompanied by a lack of tracking and management tools for machine learning models and experiments

The complexity of productionizing ML models due to the lack of integration among data pipelines, ML environments and production services

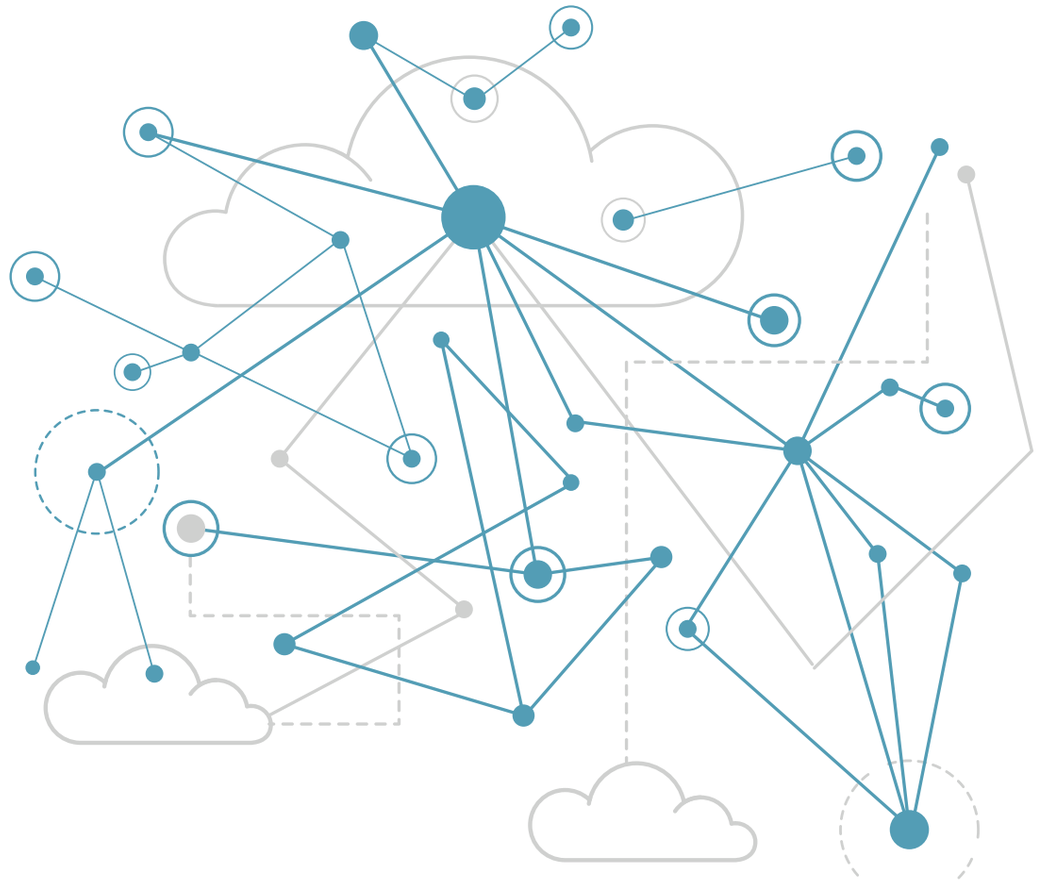
Let's look at each of these areas in turn.

The diversity and number of ML tools involved

While the traditional software development process leads to the rationalization and governance of tools and platforms used for developing and managing applications, the ML lifecycle relies on data scientists' ability to use multiple tools, whether for preparing data and training models, or deploying them for production use. Data scientists will seek the latest algorithms from the most up-to-date ML libraries and frameworks available to compare results and improve performance.



However, due to the variety of available tools and the lack of detailed tracking, teams often have trouble getting the same code to work again in the same way. Reproducing the ML workflow is a critical challenge, whether a data scientist needs to pass training code to an engineer for use in production or go back to past work to debug a problem.



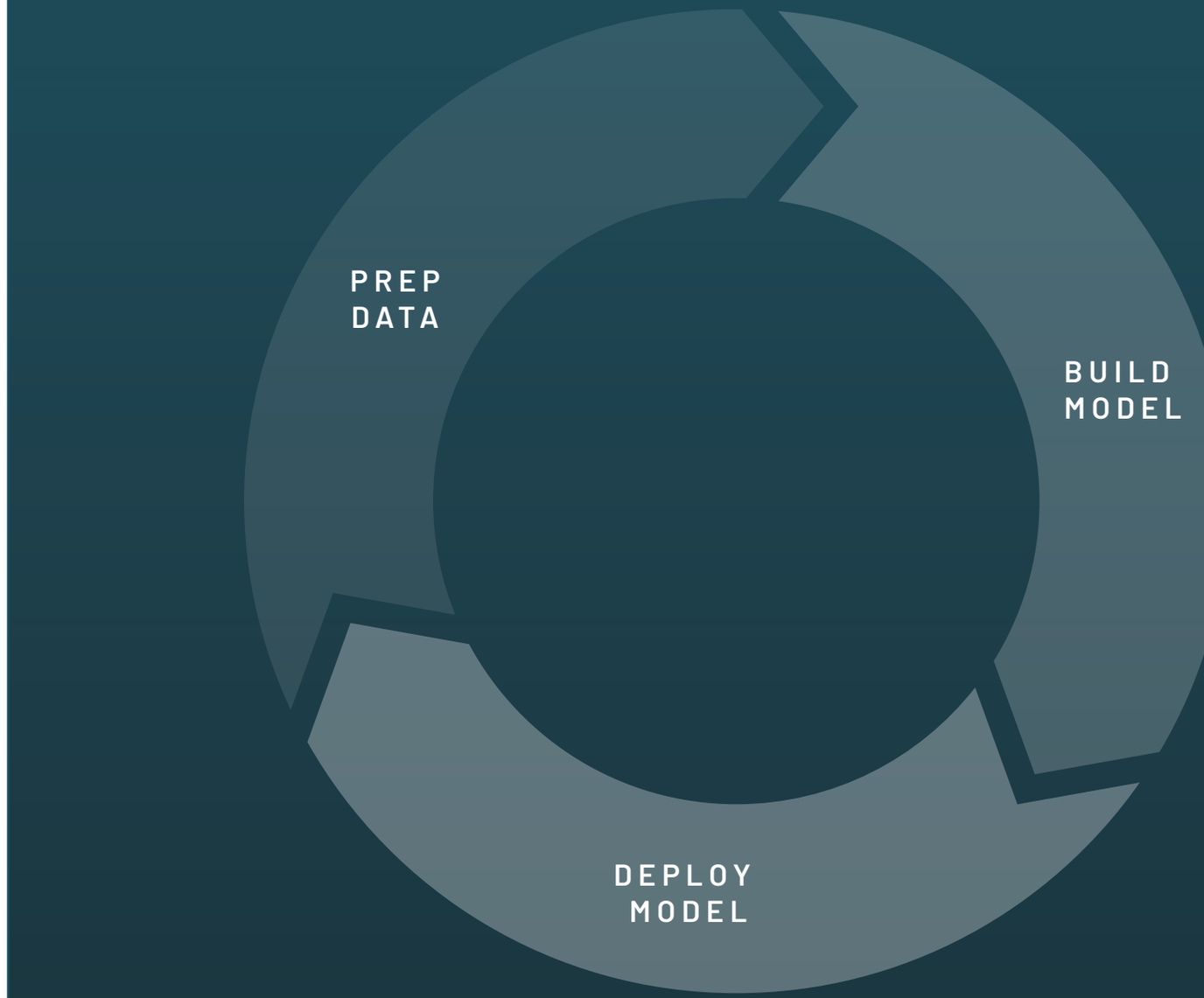
databricks

mlflow

The continuous nature of ML development

Technology never stands still. New data, algorithms, libraries and frameworks impact model performance continuously and, thus, need to be tested. Therefore, machine learning development requires a continuous approach, along with tracking capabilities to compare and reproduce results. The performance of ML models depends not only on the algorithms used, but also on the quality of the data sets and the parameter values for the models.

Whether practitioners work alone or on teams, it's still very difficult to track which parameters, code and data went into each experiment to produce a model, due to the intricate nature of the ML lifecycle itself.



The complexity of productionizing ML models

In software development, the architecture is set early on, based on the target application. Once the infrastructure and architecture have been chosen, they won't be updated or changed due to the sheer amount of work involved in rebuilding applications from scratch. Modern developments, such as the move to microservices, are making this easier, but for the most part, SDLC focuses on maintaining and improving what already exists.

With machine learning the first goal is to build a model. And keep in mind: a model's performance in terms of accuracy and sensitivity is agnostic from the deployment mode. However, models can be heavily dependent on latency, and the chosen architecture requires significant scalability based on the business application. End-to-end ML pipeline designs can be great for batch analytics and looking at streaming data, but they can involve different approaches for real-time scoring when an application is based on a microservice architecture working via REST APIs, etc.

One of today's key challenges is to effectively transition models from experimentation to staging and production — without needing to rewrite the code for production use. This is time-consuming and risky as it can introduce new bugs. There are many solutions available to productionize a model quickly, but practitioners need the ability to choose and deploy models across any platform, and scale resources as needed to manage model inference effectively on big data, in batch or real time.

CHAPTER 2: Applying Good Engineering Principles to Machine Learning

Many data science and machine learning projects fail due to preventable issues that have been resolved in software engineering for more than a decade. However, those solutions need to be adapted due to key differences between developing code and training ML models.

- **Expertise, code and data** — With the addition of data, data science and ML, code not only needs to deal with data dependencies but also handle the inherent nondeterministic characteristics of statistical modeling. ML models are not guaranteed to behave the same way when trained twice, unlike traditional code, which can be easily unit tested.
- **Model artifacts** — In addition to application code, ML products and features also depend on models that are the result of a training process. Those model artifacts can often be large — on the order of gigabytes — and often need to be served differently from code itself.
- **Collaboration** — In large organizations, models that are deployed in an application are usually not trained by the same people responsible for the deployment. Handoffs between experimentation, testing and production deployments are similar but not identical to approval processes in software engineering.

The need for standardization

Some of the world's largest tech companies have already begun solving these problems internally with their own machine learning platforms and lifecycle management tools.[2] These internal platforms have been extremely successful and are designed to accelerate the ML lifecycle by standardizing the process of data preparation, model training, and deployment via APIs built for data scientists. The platforms not only help standardize the ML lifecycle but also play a major role in retaining knowledge and best practices, and maximizing data science team productivity and collaboration, thereby leading to greater ROI.

Internally driven strategies still have limitations. First, they are limited to a few algorithms or frameworks. Adoption of new tools or libraries can lead to significant bottlenecks. Of course, data scientists always want to try the latest and the best algorithms, libraries and frameworks — the most recent versions of PyTorch, TensorFlow and so on. Unfortunately, production teams cannot easily incorporate these into the custom ML platform without significant rework. The second limitation is that each platform is tied to a specific company's infrastructure. This can limit sharing of efforts among data scientists. As each framework is so specific, options for deployment can be limited.

The question then is: Can similar benefits to these systems be provided in an open manner? This evaluation must be based on the widest possible mix of tools, languages, libraries and infrastructures. Without this approach, it will be very difficult for data scientists to evolve their ML models and keep pace with industry developments. Moreover, by making it available as open source, the wider industry will be able to join in and contribute to ML's wider adoption. This also makes it easier to move between various tools and libraries over time.

2 Facebook has implemented its FBLearner Flow platform. Uber has a service called Michelangelo. And Google has TFX.

CHAPTER 3: Introducing MLflow

MATEI ZAHARIA
Co-founder and Chief Technologist at Databricks

At Databricks, we believe that there should be a better way to manage the ML lifecycle. So in June 2018, we unveiled MLflow, an open-source machine learning platform for managing the complete ML lifecycle.

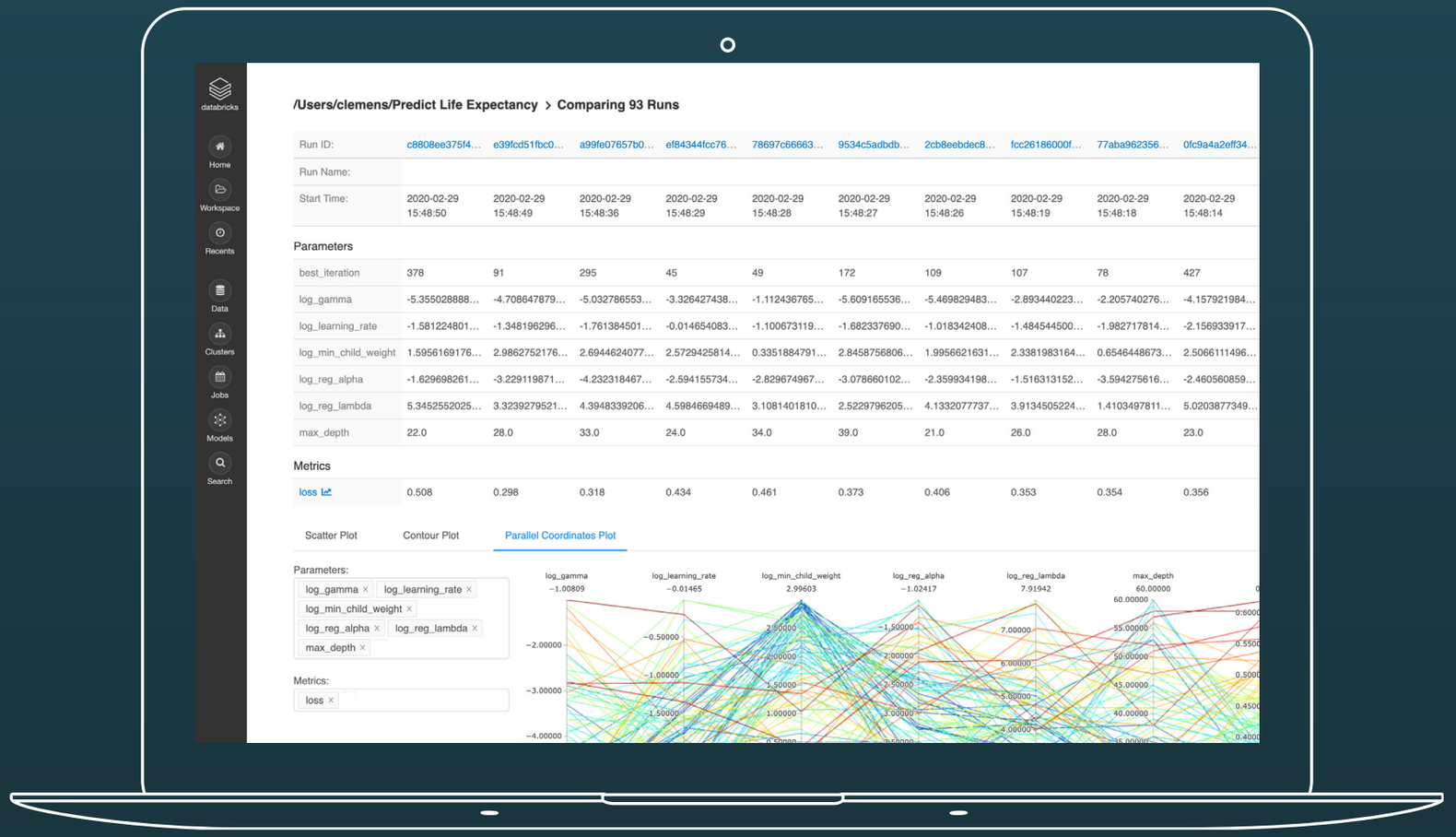
"MLflow is designed to be a cross-cloud, modular, API-first framework, to work well with all popular ML frameworks and libraries. It is open and extensible by design, and platform agnostic for maximum flexibility."

With MLflow, data scientists can now package code as reproducible runs, execute and compare hundreds of parallel experiments, and leverage any hardware or software platform for training, hyperparameter tuning and more. Also, organizations can deploy and manage models in production on a variety of clouds and serving platforms.

"With MLflow, data science teams can systematically package and reuse models across frameworks, track and share experiments locally or in the cloud, and deploy models virtually anywhere," says Zaharia. "The flurry of interest and contributions we've seen from the data science community validates the need for an open-source framework to streamline the machine learning lifecycle."

Key benefits

EXPERIMENT TRACKING As mentioned previously, getting ML models to perform takes significant trial and error, and continuous configuration, building, tuning, testing, etc. Therefore, it is imperative to allow data science teams to track all that goes into a specific run, along with the results. With MLflow, data scientists can quickly record runs and keep track of model parameters, results, code and data from each experiment, all in one place.

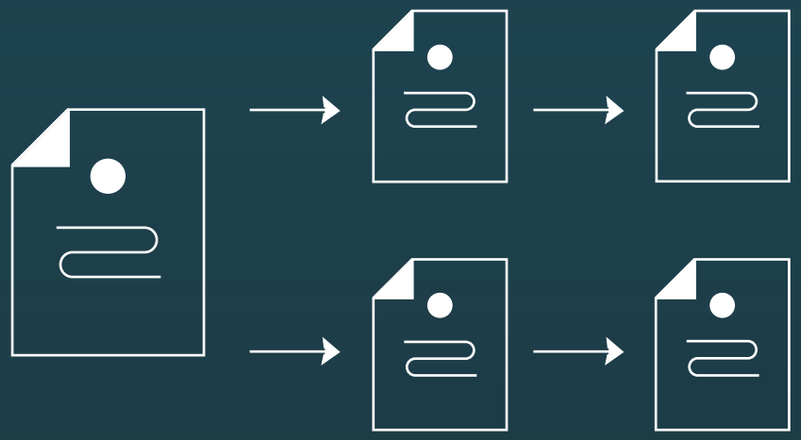


databricks

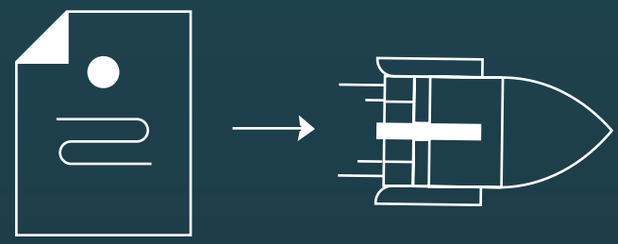
mlflow

Key benefits

REPRODUCIBLE PROJECTS The ability to reproduce a project — entirely or just parts of it — is key to data science productivity, knowledge sharing and, hence, accelerating innovation. With MLflow, data scientists can build and package composable projects, capture dependencies and code history for reproducible results, and quickly share projects with their peers.

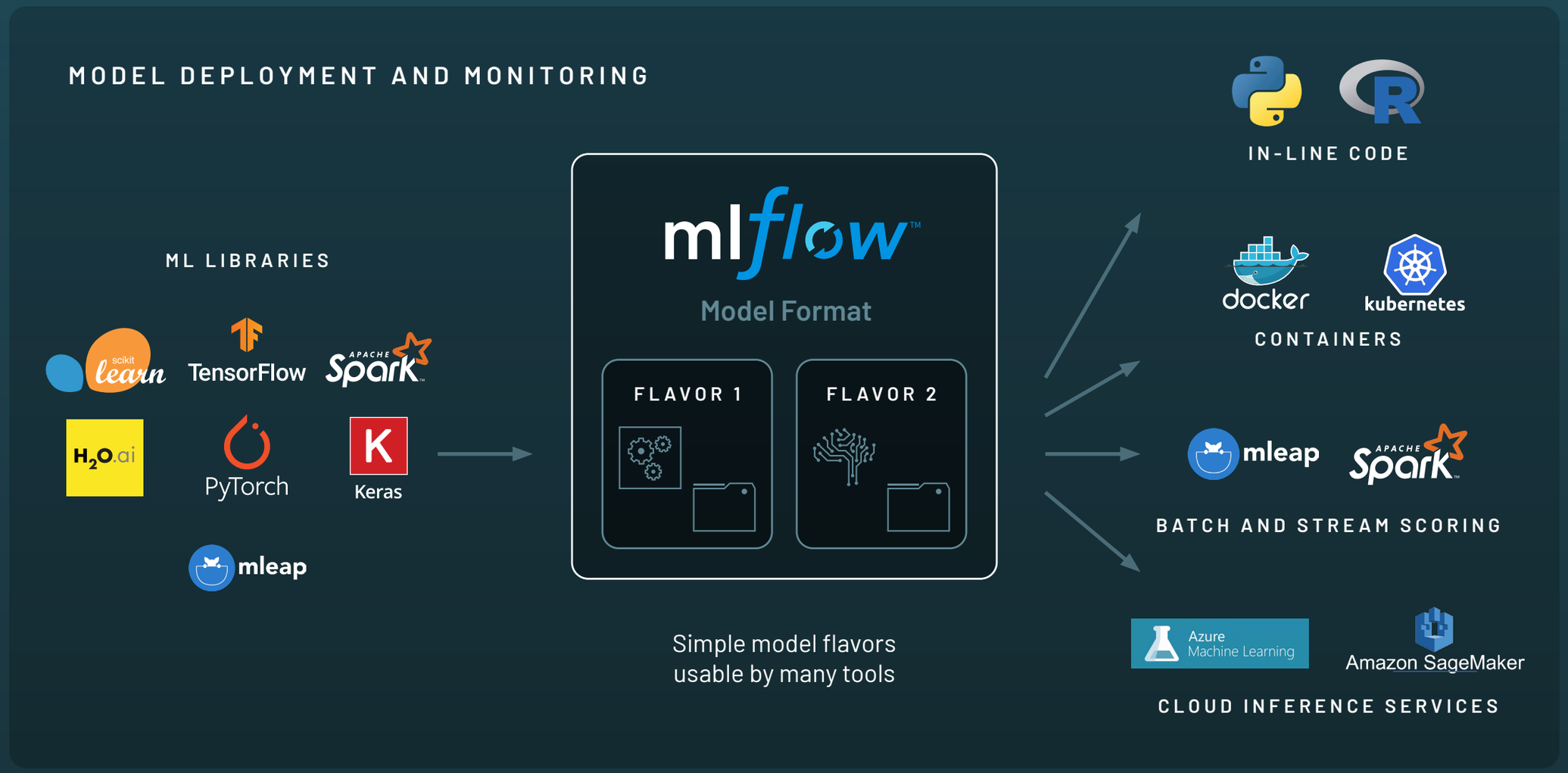


FLEXIBLE DEPLOYMENT There is virtually no limit to what machine learning can do for your business. However, there are different ways to architect ML applications for production, and various tools can be used for deploying models, which often lead to code rewrites prior to deploying ML models into production. With MLflow, your data scientists can quickly download or deploy any saved models to various platforms — locally or in the cloud — from experimentation to production.



Key benefits

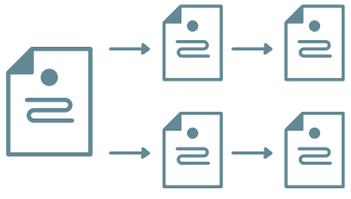
MODEL MANAGEMENT Use one central place to share ML models, collaborate on moving them from experimentation to online testing and production, integrate with approval and governance workflows, and monitor ML deployments and their performance. This is powered by the latest MLflow component, MLflow Model Registry.



Use case examples

Let's examine three use cases to explore how users can leverage some of the MLflow components.

EXPERIMENT TRACKING A European energy company is using MLflow to track and update hundreds of energy-grid models. This company's goal is to build a time-series model for every major energy producer (e.g., power plant) and consumer (e.g., factory), monitor these models using standard metrics, and combine the predictions to drive business processes, such as pricing. Because a single team is responsible for hundreds of models, possibly using different ML libraries, it's important to have a standard development and tracking process. The team has standardized on Jupyter notebooks for development, MLflow Tracking for metrics, and Databricks Jobs for inference.



REPRODUCIBLE PROJECTS An online marketplace is using MLflow to package deep learning jobs using Keras and run them in the cloud. Each data scientist develops models locally on a laptop using a small data set, checks them into a Git repository with an MLproject file, and submits remote runs of the project to GPU instances in the cloud for large-scale training or hyperparameter search. Using MLflow Projects makes it easy to create the same software environment in the cloud and share project code among data scientists.

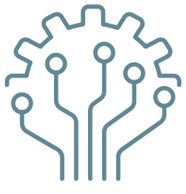
MODEL PACKAGING An e-commerce site's data science team is using MLflow Model Registry to package recommendation models for use by application engineers. This presents a technical challenge because the recommendation application includes both a standard, off-the-shelf recommendation model and custom business logic for pre- and post-processing. For example, the application might include custom code to ensure the recommended items are diverse. This business logic needs to change in sync with the model, and the data science team wants to control both the business logic and the model, without having to submit a patch to the web application each time the logic has to change. Moreover, the team wants to A/B test distinct models with distinct versions of the processing logic. The solution was to package both the recommendation model and the custom logic using the `python_function` flavor in an MLflow Model, which can then be deployed and tested as a single unit.

Open and extensible by design

Since we unveiled and open sourced MLflow in June 2018 at the Spark + AI Summit in San Francisco, community engagement and contributions have led to an impressive array of new features and integrations:

SUPPORT FOR MULTIPLE PROGRAMMING LANGUAGES

To give developers a choice, MLflow supports R, Python, Java and Scala, along with a REST server interface that can be used from any language.



INTEGRATION WITH POPULAR ML LIBRARIES AND FRAMEWORKS

MLflow has built-in integrations with the most popular machine learning libraries — such as scikit-learn, TensorFlow, Keras, PyTorch, H2O, and Apache Spark™ MLlib — to help teams build, test and deploy machine learning applications.

CROSS-CLOUD SUPPORT

Organizations can use MLflow to quickly deploy machine learning models to multiple cloud services, including Databricks, Azure Machine Learning and Amazon SageMaker, depending on their needs. MLflow leverages AWS S3, Google Cloud Storage and Azure Data Lake Storage, allowing teams to easily track and share artifacts from their code.

Rapid community adoption

1.8M
monthly downloads

200+
code contributors

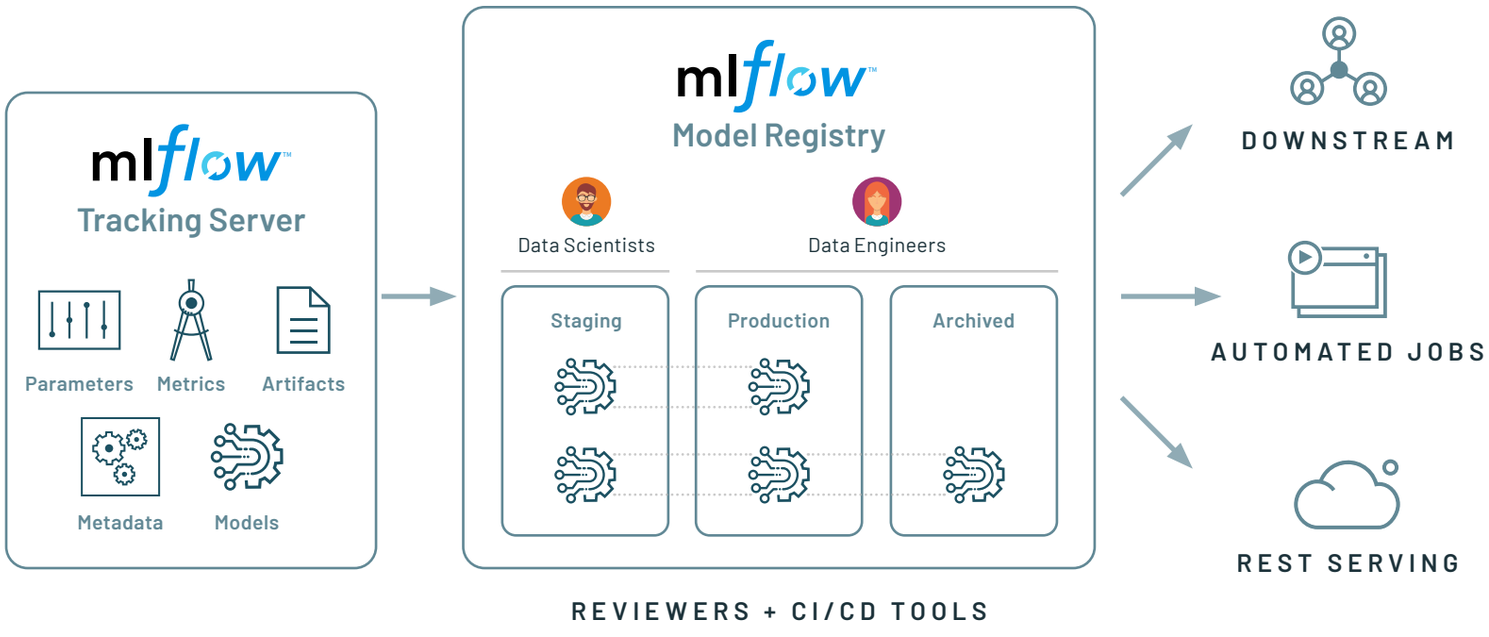
100+
contributing organizations

Organizations using and contributing to MLflow
Source: mlflow.org

CHAPTER 4: A Closer Look at MLflow Model Registry

MLflow originally introduced the ability to track metrics, parameters and artifacts as part of experiments, package models and reproducible ML projects, and deploy models to batch or to real-time serving platforms.

The latest MLflow component — MLflow Model Registry — builds on MLflow's original capabilities to provide organizations with one central place to share ML models, collaborate on moving them from experimentation to testing and production, and implement approval and governance workflows.



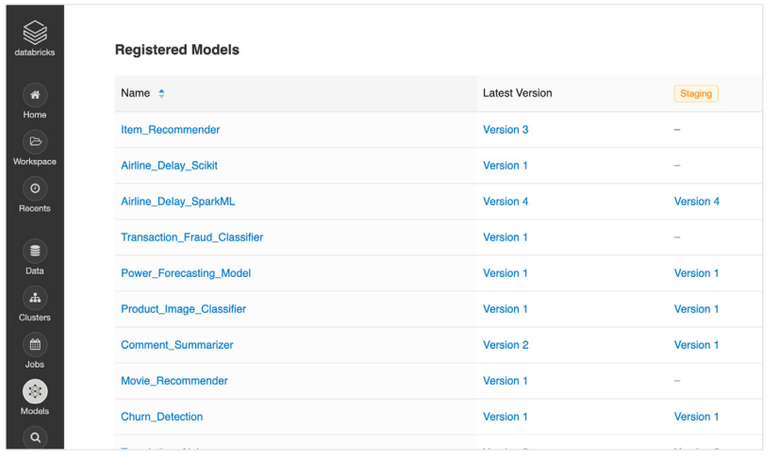
The Model Registry gives MLflow users new tools for sharing, reviewing and managing ML models throughout their lifecycle

The MLflow Model Registry complements the MLflow offering and is designed to help organizations implement good engineering principles with machine learning initiatives, such as collaboration, governance, reproducibility and knowledge management. The next few pages highlight some of the key features of this new component.

One hub for managing ML models collaboratively

Building and deploying ML models is a team sport. Not only are the responsibilities along the machine learning model lifecycle often split across multiple people (e.g., data scientists train models whereas production engineers deploy them), but also at each lifecycle stage, teams can benefit from collaboration and sharing (e.g., a fraud model built in one part of the organization could be reused in others).

MLflow facilitates sharing of expertise and knowledge across teams by making ML models more discoverable and providing collaborative features to jointly improve on common ML tasks. Simply register an MLflow model from your experiments to get started. The MLflow Model Registry will then let you track multiple versions of the model and mark each one with a lifecycle stage: development, staging, production or archived.

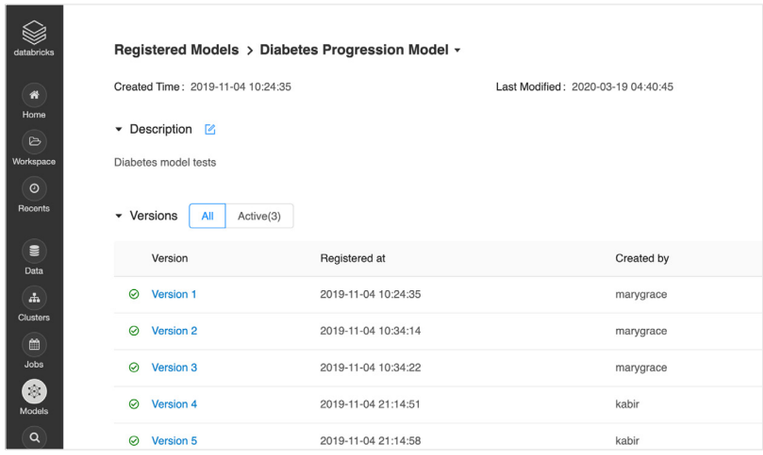


Sample machine learning models displayed via the MLflow Model Registry dashboard

Flexible CI/CD pipelines to manage stage transitions

MLflow lets you manage your models' lifecycles either manually or through automated tools. Analogous to the approval process in software engineering, users can manually request to move a model to a new lifecycle stage (e.g., from staging to production), and review or comment on other users' transition requests.

Alternatively, you can use the Model Registry's API to plug in continuous integration and deployment (CI/CD) tools, such as Jenkins, to automatically test and transition your models. Each model also links to the experiment run that built it — in MLflow Tracking — to let you easily review models.

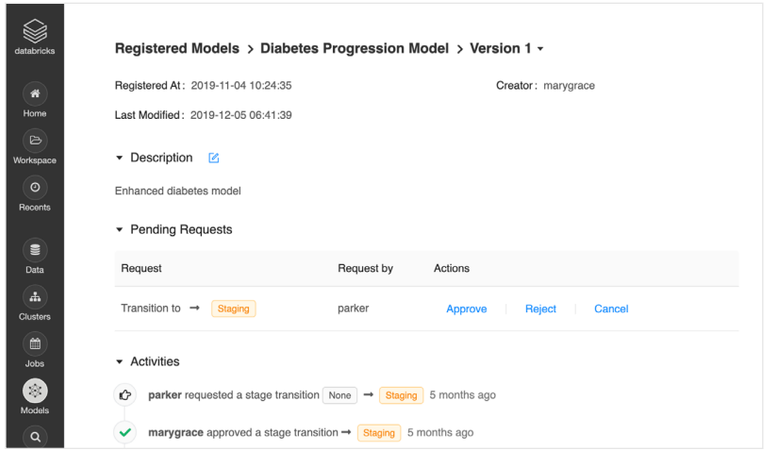


The machine learning model page view in MLflow, showing how users can request and review changes to a model's stage

Visibility and governance for the full ML lifecycle

In large enterprises, the number of ML models that are in development, staging and production at any given point in time may be in the hundreds or thousands. Having full visibility into which models exist, what stages they are in and who has collaborated on and changed the deployment stages of a model allows organizations to better manage their ML efforts.

MLflow provides full visibility and enables governance by keeping track of each model's history and managing who can approve changes to the model's stages.

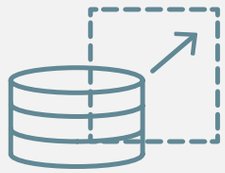


Identify versions, stages and authors of each model

CHAPTER 5: Making Organizations Successful with ML

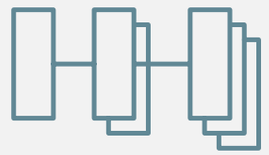
Standardizing the ML lifecycle with MLflow is a great step to ensure that data scientists can share and track experiments, compare results, reproduce runs and productionize faster.

In addition to increasing data science team productivity and collaboration and applying good engineering practices to machine learning, organizations also need to do the following:



Reliably ingest, ETL and catalog big data

Work with state-of-the-art ML frameworks and tools

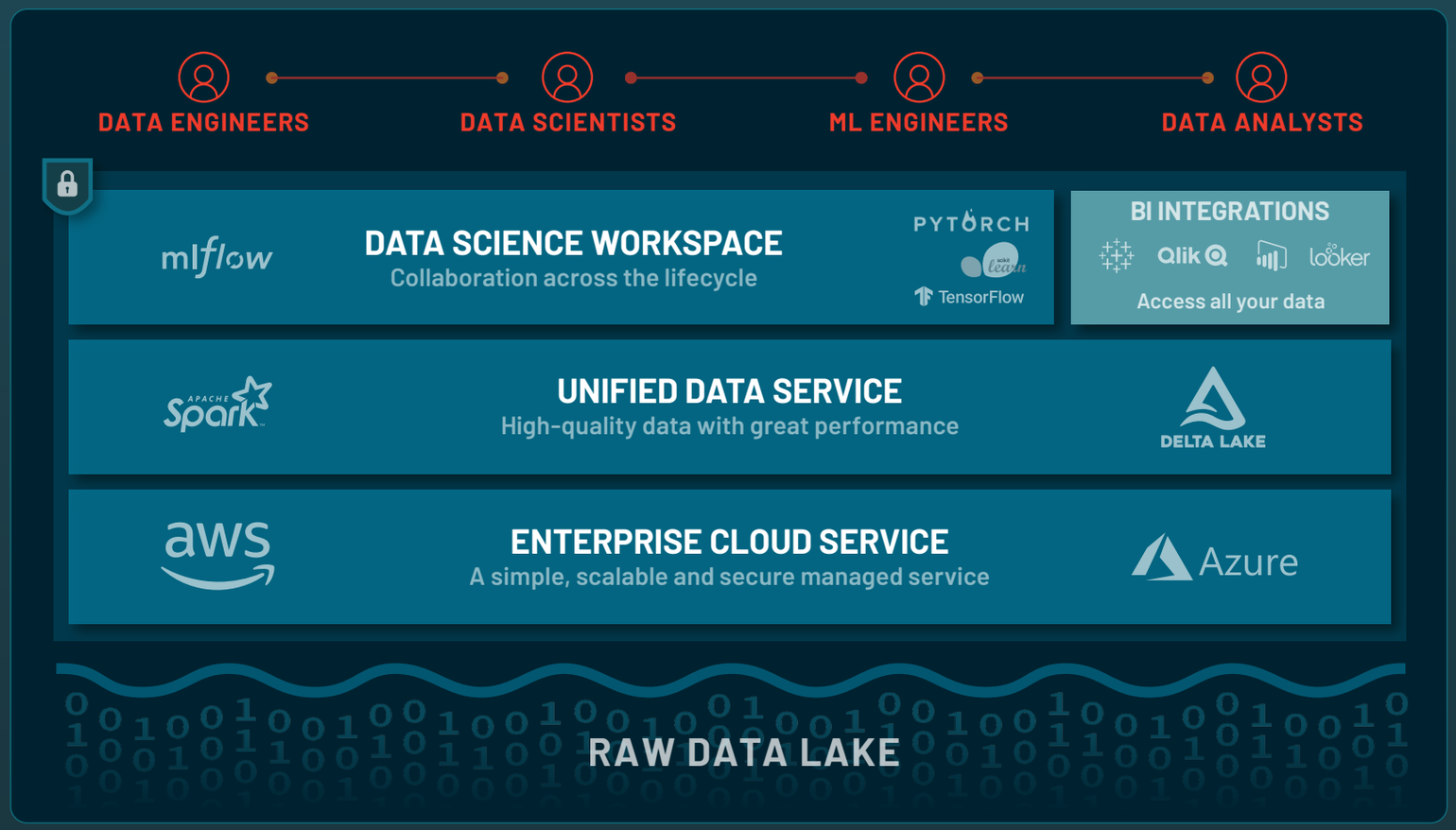


Easily scale compute from single to multi-node

Databricks excels at all the above. Learn more at databricks.com

CHAPTER 6: Introducing the Unified Data Analytics Platform

Databricks accelerates innovation by unifying data science, engineering and business. Through a fully managed, cloud-based service built by the original creators of Apache Spark, Delta Lake and MLflow, the Databricks Unified Data Analytics Platform lowers the barrier for enterprises to innovate with AI and accelerates their innovation.



Data engineering

Speed up the preparation of high-quality data, essential for best-in-class ML applications, at scale

Data science

Collaboratively explore large data sets, build models iteratively and deploy across multiple platforms

Providing managed MLflow on Databricks

MLflow is natively integrated with the Databricks Unified Data Analytics Platform so that ML practitioners and engineers can benefit from out-of-the-box tracking, packaging, deployment and management capabilities for ML models with enterprise reliability, security and scale.

By using MLflow as part of Databricks, data scientists can:

WORKSPACES

Benefit from a streamlined experiment tracking experience with Databricks Workspace and collaborative Notebooks

BIG DATA SNAPSHOTS

Track large-scale data that fed the models, along with all the other model parameters, then reproduce training runs reliably

JOBS

Easily initiate jobs remotely, from an on-premises environment or from Databricks notebooks

SECURITY

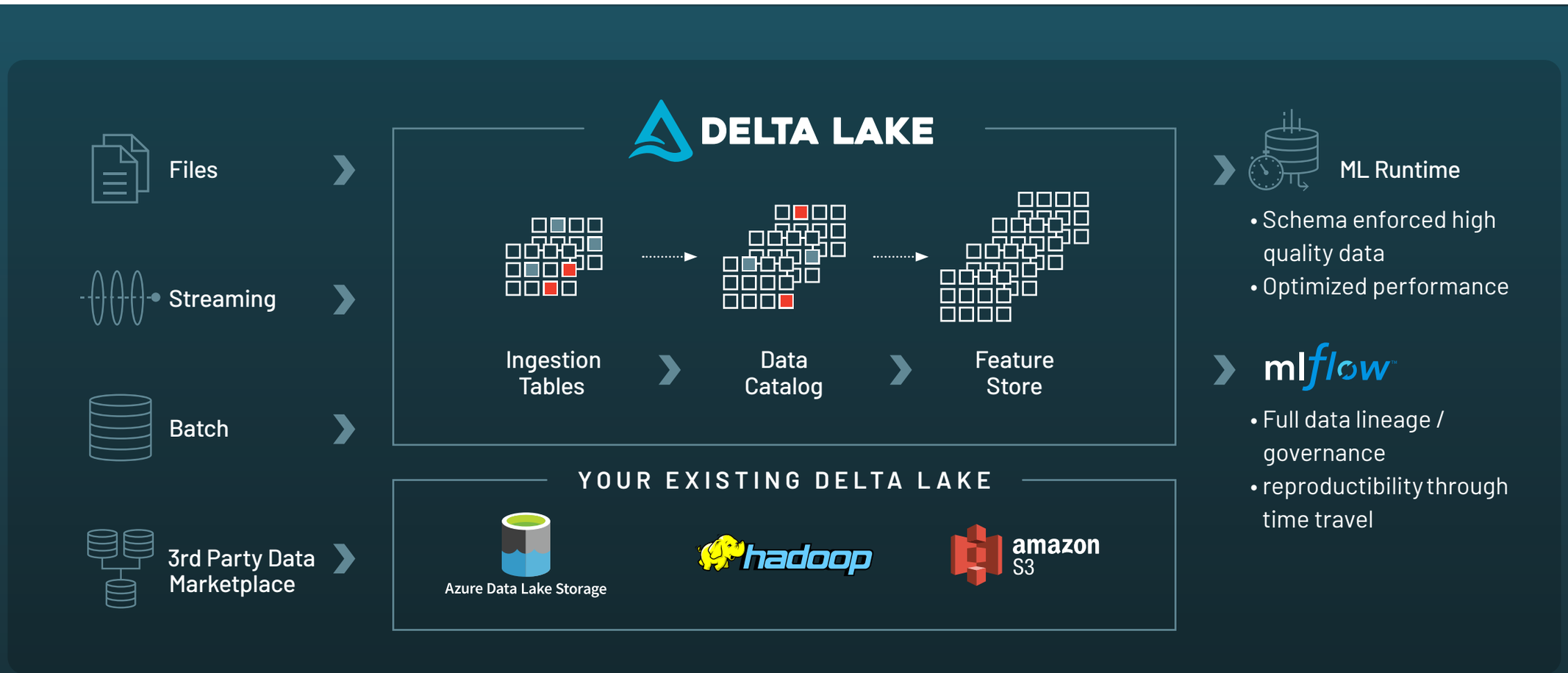
Take advantage of one common security model for the entire machine learning lifecycle

Read our blog to learn more about these integrations.

Getting data ready for ML with Delta Lake

Delta Lake is a storage layer that brings reliability to data lakes. Delta Lake provides ACID transactions and scalable metadata handling, and it unifies streaming and batch data processing. Delta Lake runs on top of your existing data lake and is fully compatible with Apache Spark APIs.

By using Delta Lake, data engineers and data scientists can keep track of data used for model training.



Ready-to-use ML environments

Databricks Runtime for Machine Learning provides data scientists and ML practitioners with on-demand access to ready-to-use machine learning clusters that are preconfigured with the latest and most popular machine learning frameworks, including TensorFlow, Keras, PyTorch, scikit-learn, XGBoost and Horovod.

By using the Databricks Runtime for ML, data scientists can get to results faster with one-click access to ML clusters, optimized performance on popular ML algorithms, and simplified distributed deep learning on Horovod and GPUs. It also supports Conda for further customization.

PACKAGES AND OPTIMIZES MOST COMMON ML FRAMEWORKS

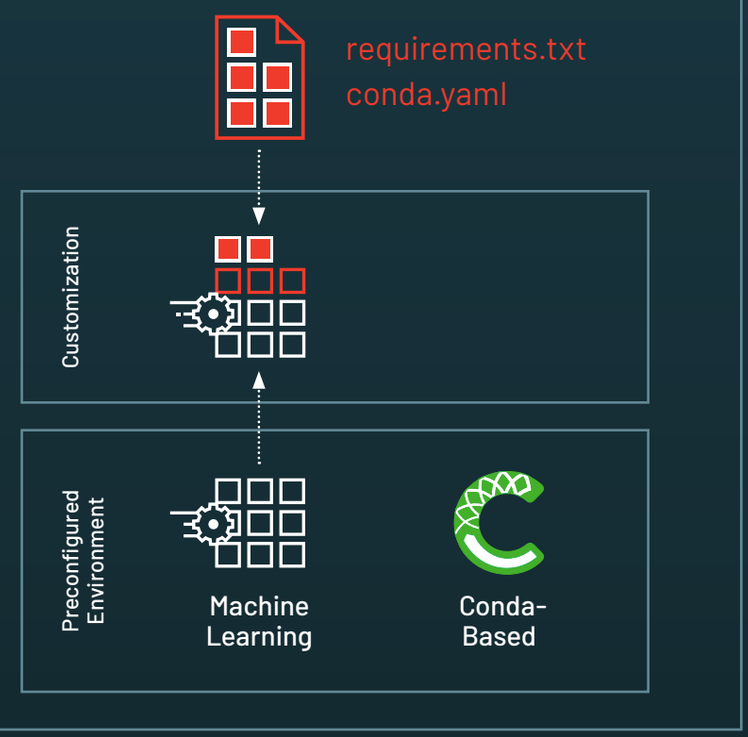
BUILT-IN OPTIMIZATION FOR DISTRIBUTED DEEP LEARNING

Distribute and Scale any Single-Machine ML Code to thousands of machines

BUILT-IN AUTOML AND EXPERIMENT TRACKING

Auto ML and Tracking / Visualizations with MLflow

CUSTOMIZED ENVIRONMENTS USING CONDA



CHAPTER 7: Standardizing the Machine Learning Lifecycle on Databricks

CHAPTER 8: Getting Started

Take the next step toward standardizing your ML lifecycle — test drive MLflow and the Databricks Unified Data Analytics Platform.

START YOUR FREE TRIAL

REQUEST A PERSONALIZED DEMO

LEARN MORE

JOIN THE COMMUNITY

CHAPTER 8: Comparison Matrix

	OPEN SOURCE MLFLOW	MANAGED MLFLOW ON DATABRICKS
EXPERIMENT TRACKING		
MLflow Tracking API	✅	✅
MLflow Tracking Server	✅ Self-hosted	✅ Fully managed
Notebook Integration	❌	✅
Workspace Integration	❌	✅
REPRODUCIBLE PROJECTS		
MLflow Projects	✅	✅ With remote execution
GitHub and Conda Integration	✅	✅
Scalable Cloud/Clusters for Project Runs	❌	✅
MODEL MANAGEMENT		
MLflow Model Registry	✅	✅
Model Versioning	✅	✅
Stage Transitions and Comments	✅	✅
CI/CD Workflow Integration	✅	✅
Model Stage	✅	✅
FLEXIBLE DEPLOYMENT		
MLflow Models	✅	✅
Built-In Batch Inference	❌	✅
Built-In Streaming Analytics	❌	✅
SECURITY AND MANAGEMENT		
High Availability	❌	✅
Automated Updates	❌	✅
Role-Based Access Control	❌	✅