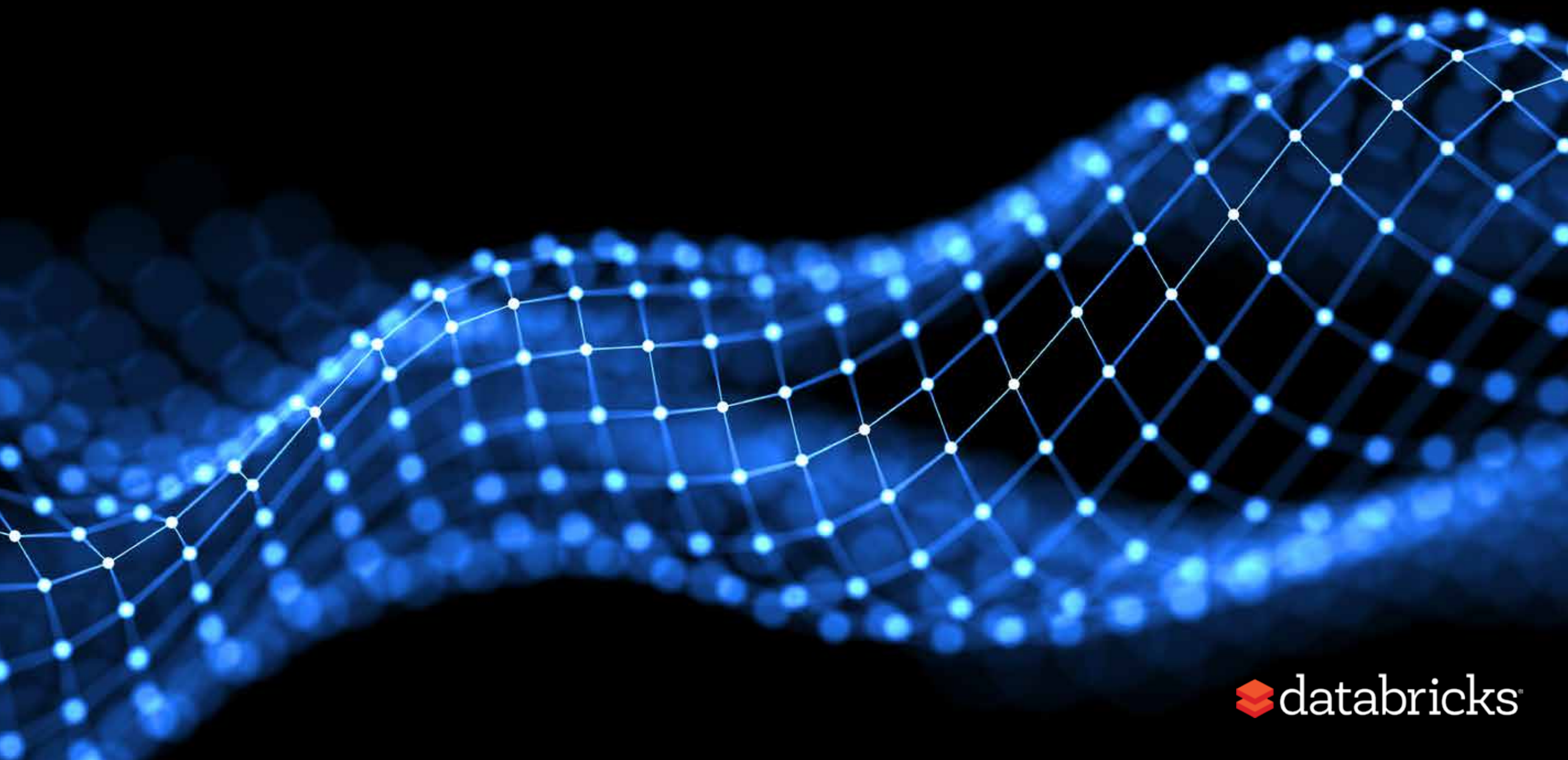
Standardizing the Machine Learning Lifecycle



databricks

PREFACE

Successfully building and deploying a machine-learning model can be difficult to do once. Enabling other data scientists (or yourself) to reproduce your pipeline, compare the results of different versions, track what's running where, and redeploy and rollback updated models, is much harder.

In this eBook, we'll explore in greater depth what makes the ML lifecycle so challenging compared to the traditional software-development lifecycle, and share the Databricks approach to addressing these challenges.

Table of Contents

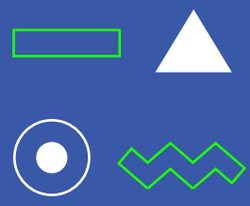
The Machine Learning Lifecycle Challenges

Building and deploying a machine-learning model can be difficult to accomplish. Enabling other data scientists – or even yourself – to reproduce your pipeline is equally challenging. Moreover, doing so can impact your data science team's productivity, leading to a significant waste of time and resources. How many times have you or your peers had to discard previous work because it was either not documented properly or, perhaps, too difficult to replicate?

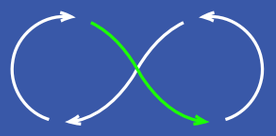
Getting models up to speed in the first place is significant enough that it can be easy to overlook long-term management. What does this involve in practice? In essence, we have to compare the results of different versions of ML models to track what's running where, and to redeploy and rollback updated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

databricks mlflow

This represents a serious shift, and challenges compare to those of 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



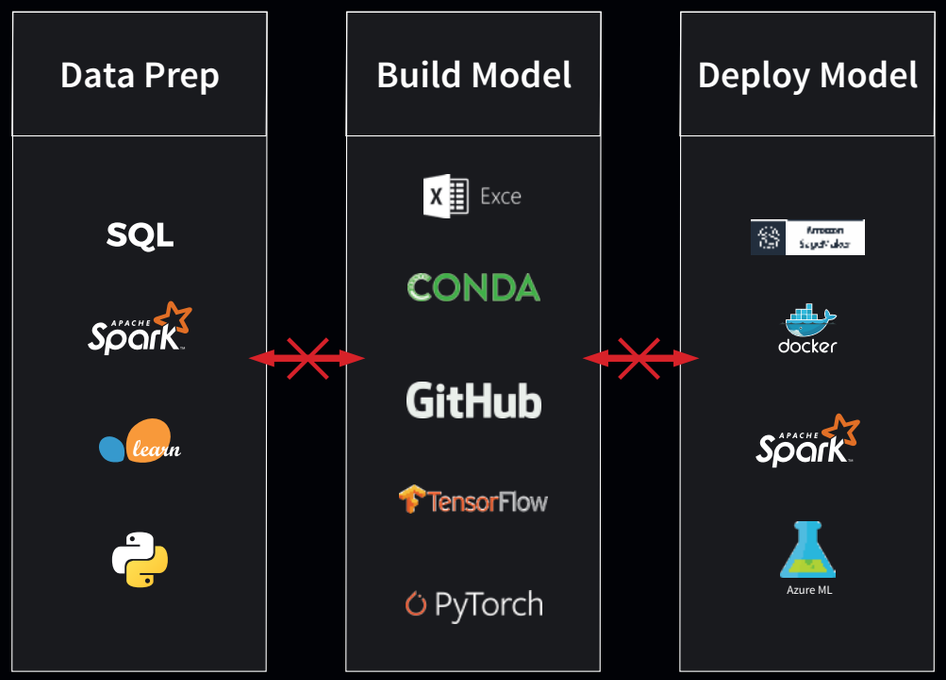
The continuous nature of ML development, coupled with a lack of tracking and management tools for machine learning models and experiments

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

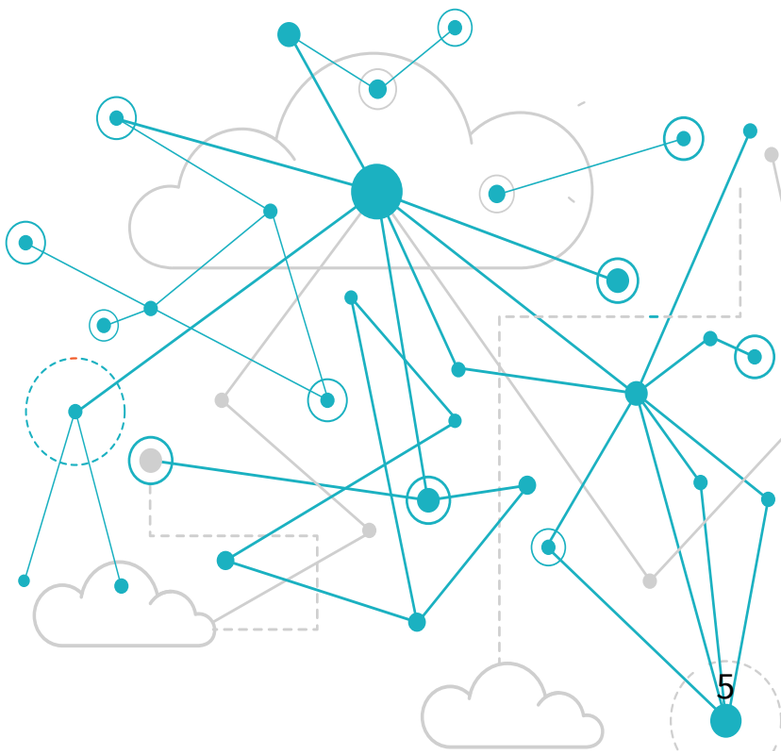
Let's look at each of these areas in turn.

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.



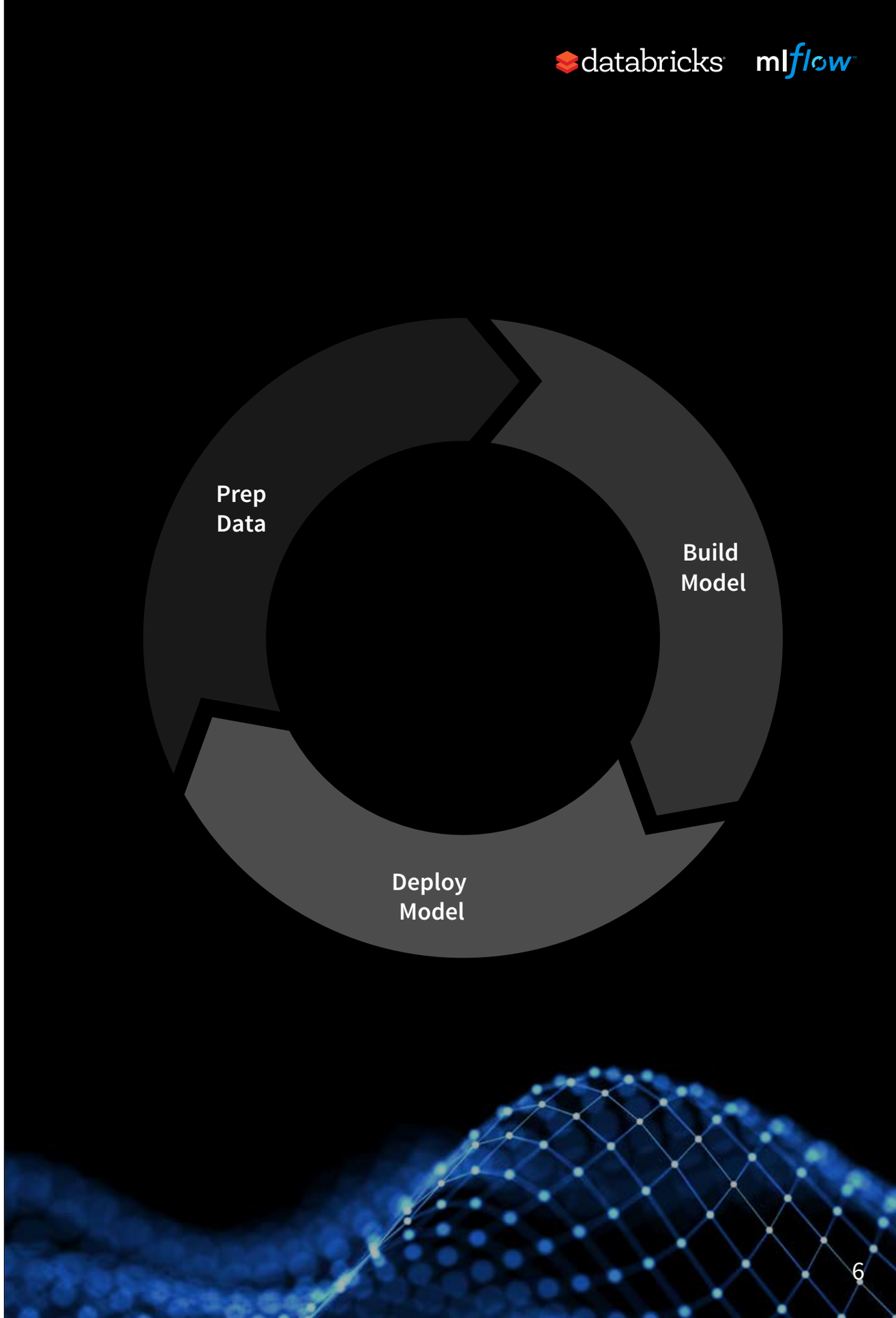
However, due to the variety of available tools and without detailed tracking, teams often have trouble getting the same code to work again in the same ways. Reproducing the ML workflow is a critical challenge, whether a data scientist needs to pass training code to an engineer for use in production, or go back to past work to debug a problem.



The Continuous Nature of ML Development

Nothing ever stands still. New data, algorithms, libraries, and frameworks impact model performance continuously and thus need to be tested as mentioned before. Therefore, machine learning development requires a continuous approach, along with tracking capabilities to compare and reproduce results. ML models' performance depends not only on the algorithms used, but also on the quality of the data sets and the parameters' values for the models.

Whether practitioners work alone or on teams, it is 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, and the lack of standardization of ML tools and processes.



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.

Conversely, in machine learning, the first goal is to build a model. For example, a model's performance in terms of accuracy and sensitivity is agnostic from the deployment mode. However, applications 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 microservices architecture working via REST APIs, etc.

Therefore, one of the key challenges today is to effectively transition models from experimentation to production, without necessarily rewriting 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.

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[1]. 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 a greater ROI.

There are still limitations to internally driven strategies. 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, e.g., the latest 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.

[1] Facebook has implemented its FBLearner platform, Uber has a service called Michelangelo, and Google has TFX.

The Need for Standardization

The question, then, is, can we provide similar benefits 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.

Introducing MLflow

At Databricks, we believe that there should be a better way to manage the ML lifecycle. Therefore, last June, we unveiled MLflow, an open source framework designed to manage the complete ML lifecycle.

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.

"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."

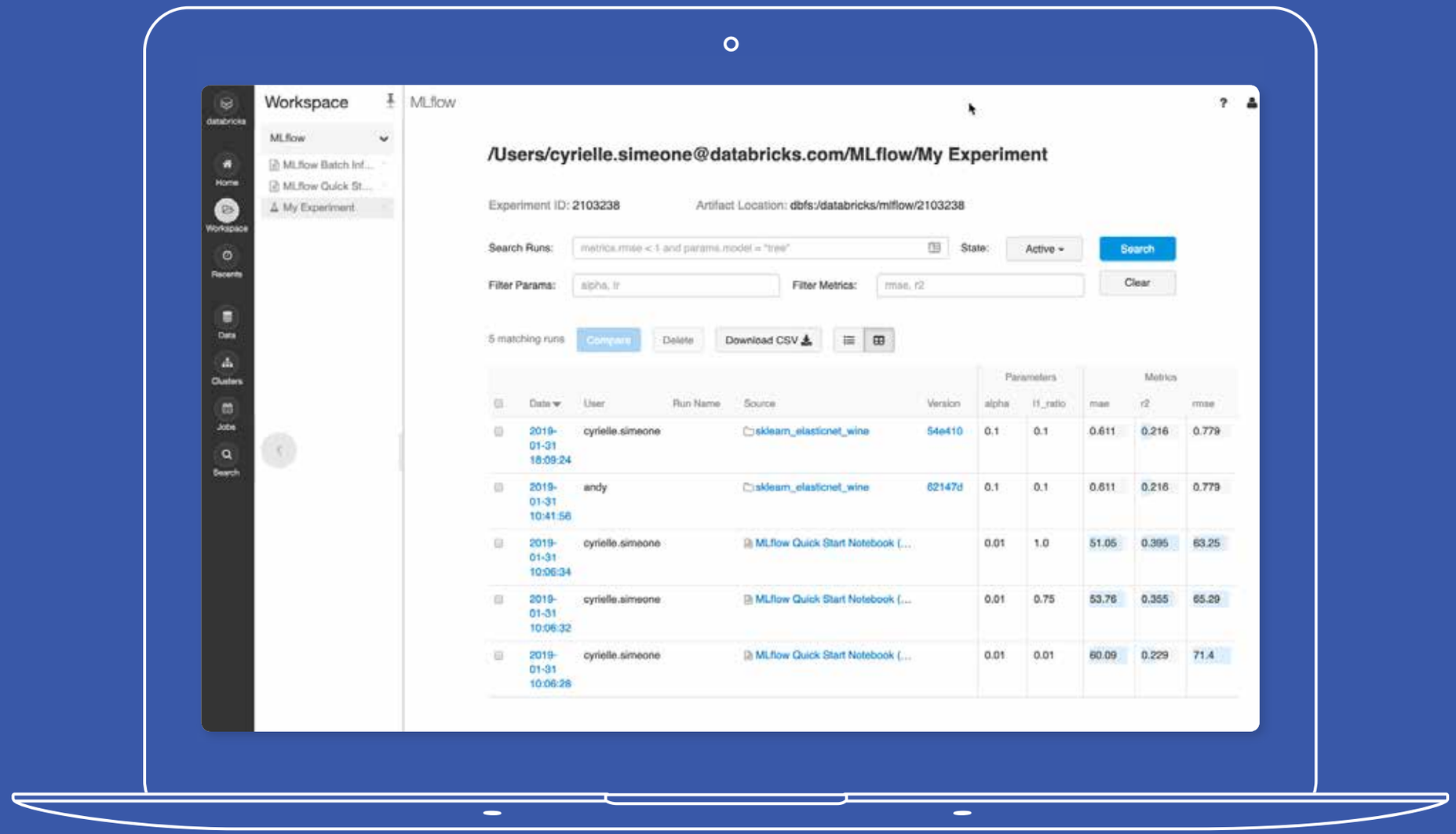
Matei Zaharia, co-founder and Chief Technologist at Databricks.

"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

Experiments Tracking

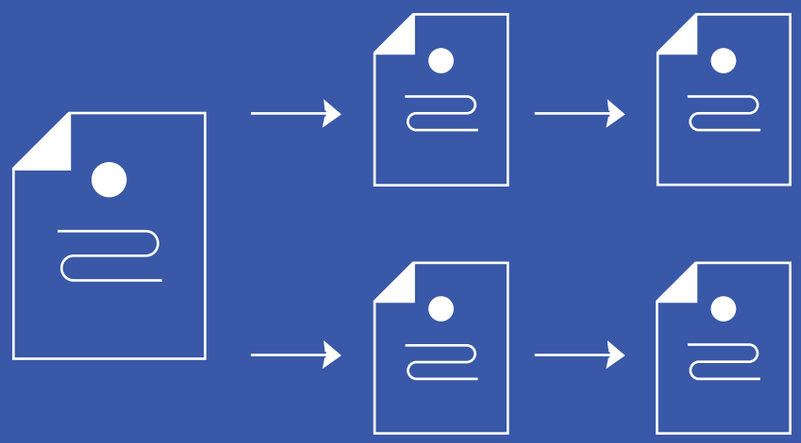
As mentioned previously, getting ML models to perform takes significant trial and error, and continuous configuration, building, tuning, testing, etc. Therefore, it is imperative to allow data-scientist teams to track all that goes into a specific run, along with the results. With MLflow, data scientists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



Key Benefits

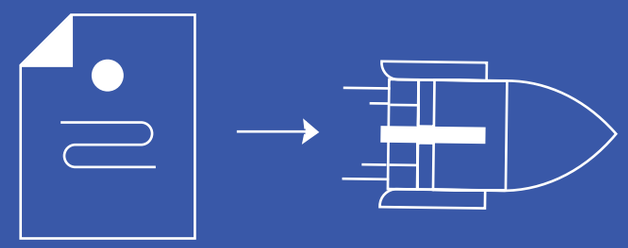
Reproducible Projects

As discussed earlier, 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.

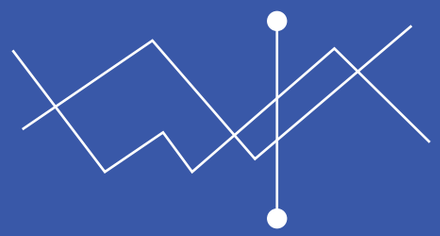


Model Deployment

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 leads 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.

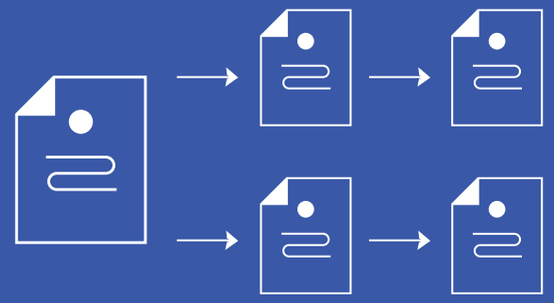


Use-Case Examples



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

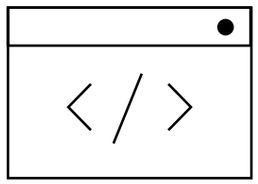
An online marketplace is using MLflow to package deep-learning jobs using Keras and run them in the cloud. Each data scientist develops models locally on a laptop using a small dataset, checks them into a Git repository with an MLproject file, and submits remote runs of the project to GPU instances in 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 Models to package recommendation models for use by application engineers. The technical challenge here was that 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 make sure that 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.

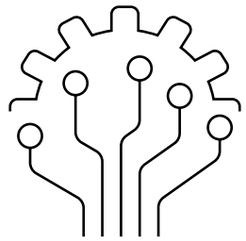
Open and Extensible by Design

Since we unveiled and open source the code last 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:



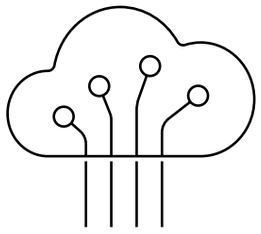
SUPPORT FOR MULTIPLE PROGRAMMING LANGUAGES

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

416K
monthly downloads

74
code contributors

40
contributing organizations

"Brandless was looking for a model management and deployment system that required limited engineering support and allowed for high flexibility. We began as an alpha tester and now run our full production recommendation system through MLflow."

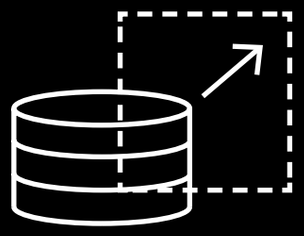
Adam Barnhard, Head of Data,

BRANDLESS™

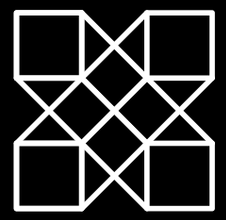
Making Organizations Successful with ML

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

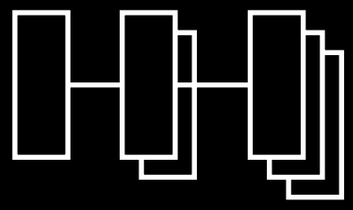
To accelerate innovation with Machine Learning, customers also need to:



Ingest, ETL, and Store Big Data



Use State-of-the-Art ML Frameworks

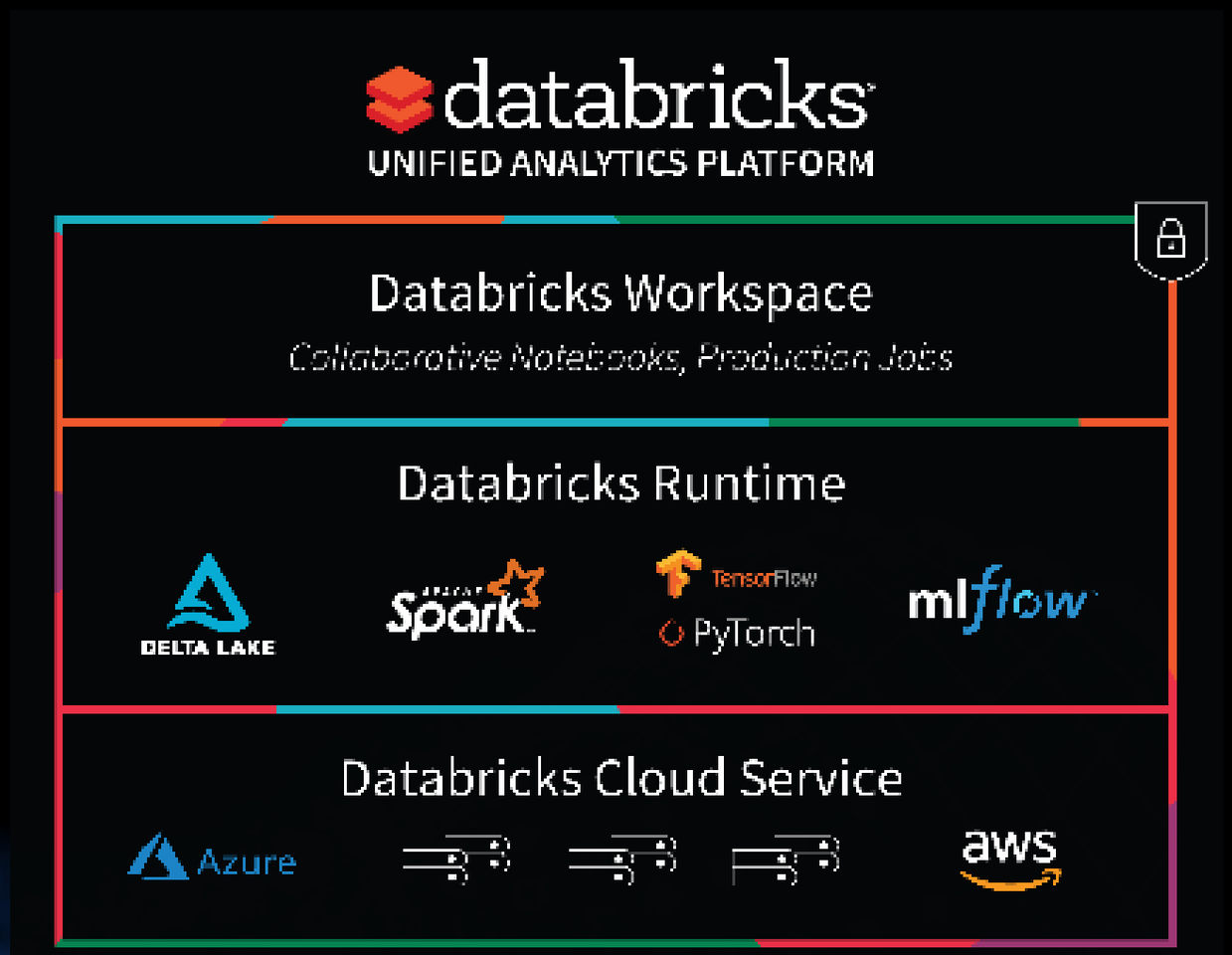


Scale Compute from Single to Multi-Nodes

databricks mlflow

Introducing Databricks Unified 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, the Databricks Unified Analytics Platform lowers the barrier for enterprises to innovate with AI and accelerates their innovation.



> “ Databricks lets us focus on business problems and make data science very simple. ”
>
> DAN MORRIS
> Senior Director of Product Analytics at broadcast giant Viacom, which used Databricks to help it identify video quality issues, increase customer loyalty, and improve advertising performance.

Data Engineering

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

Data Science

Collaboratively explore large datasets, build models iteratively and deploy across multiple platforms.

Providing Managed MLflow on Databricks

By using managed MLflow on Databricks, practitioners can benefit from out-of-the-box and seamless models tracking, packaging, and deployment capabilities with enterprise reliability, security and scale.

Workspaces
Collaboratively track and organize experiments from the Databricks Workspace

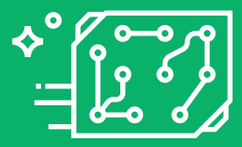
Big Data Snapshots
Track large-scale data sets that fed models with Delta Lake snapshots

Jobs
Execute runs as Databricks jobs remotely or directly from Databricks notebooks

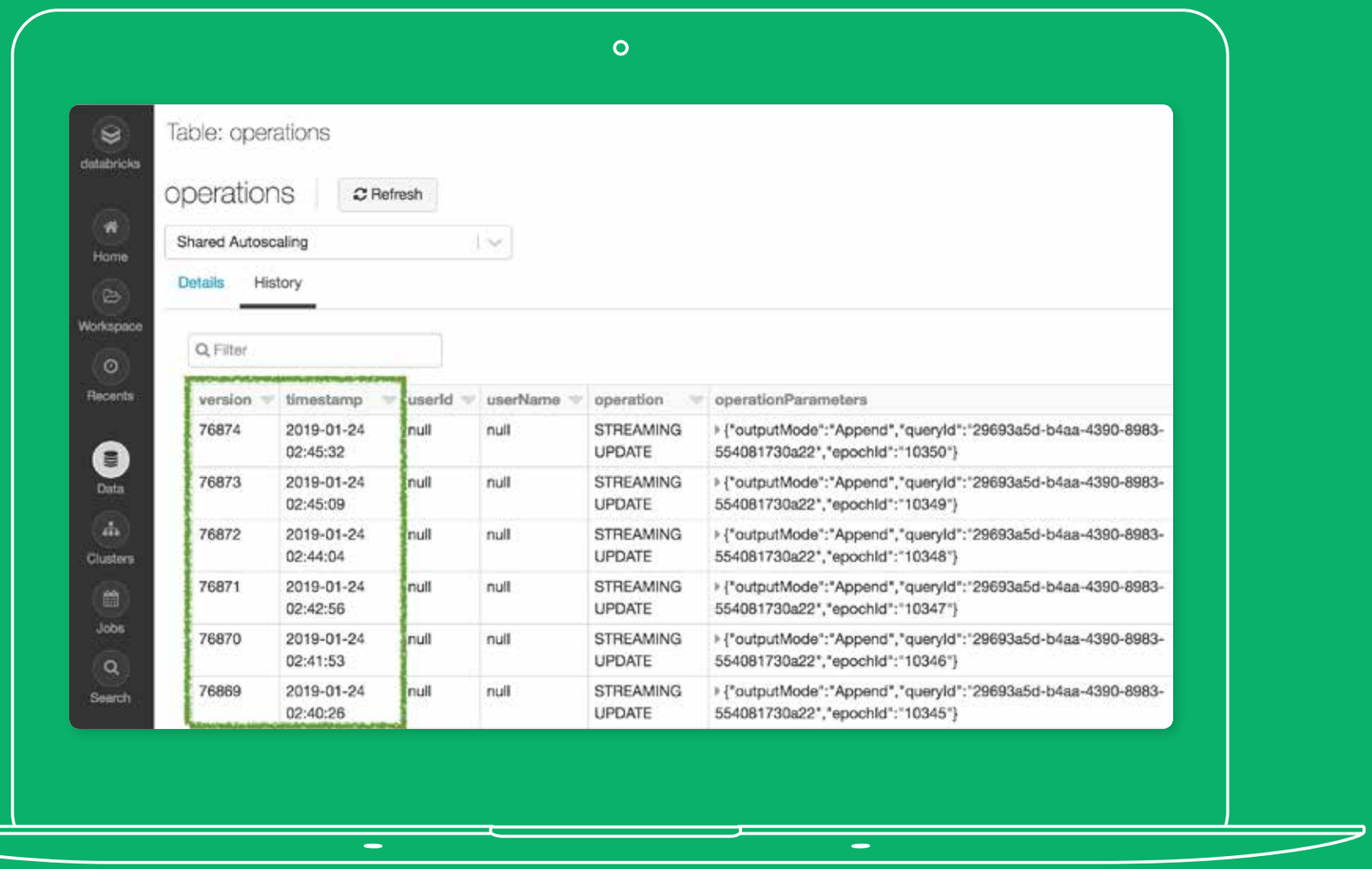
Security
Take advantage of one common security model for the entire ML lifecycle

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, scalable metadata handling, and unifies streaming and batch data processing. Delta Lake runs on top of your existing data lake and is fully compatible with Apache Spark APIs.



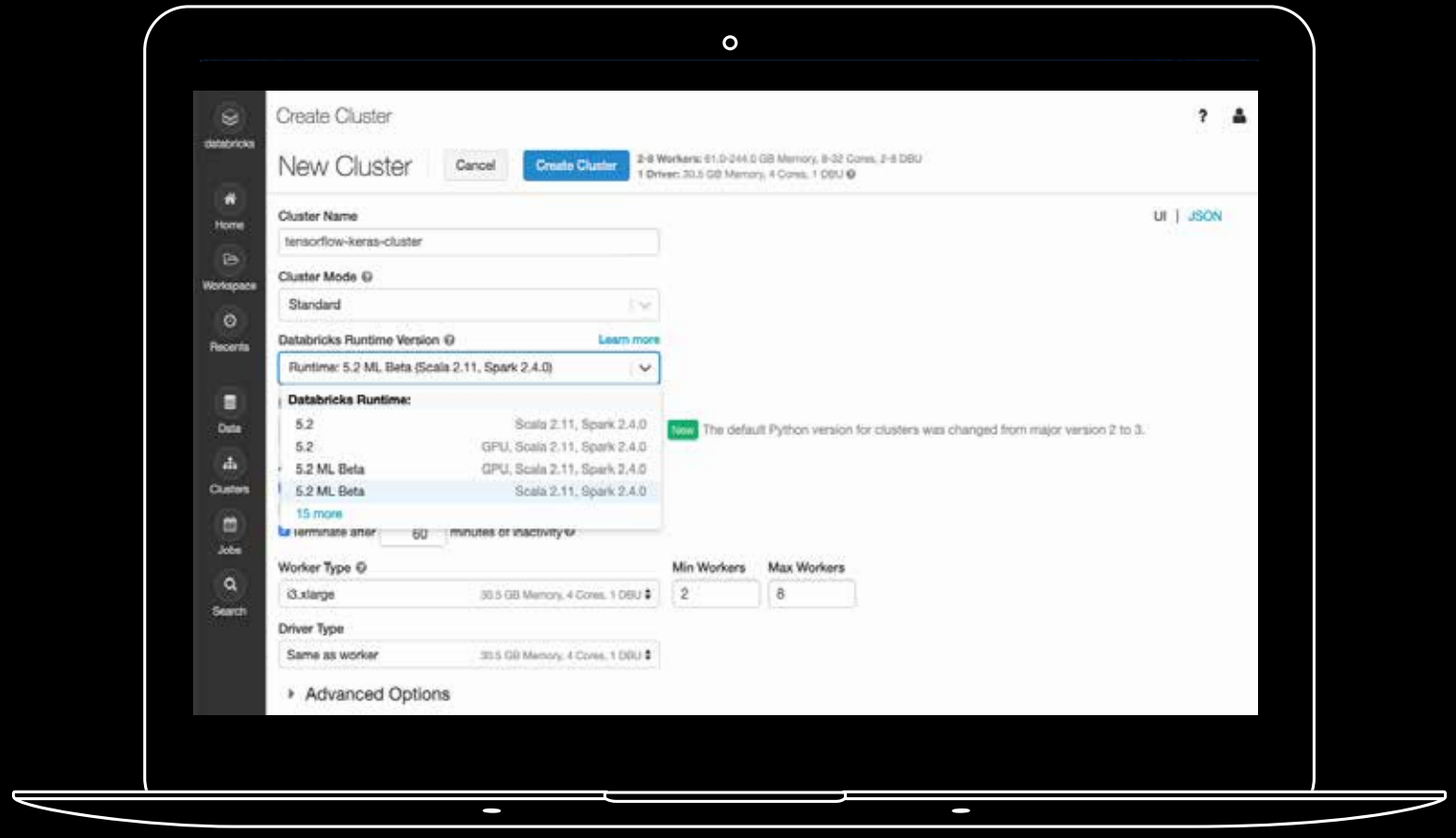
By using Delta Lake, data engineers and data scientists can keep track of data used for model training.



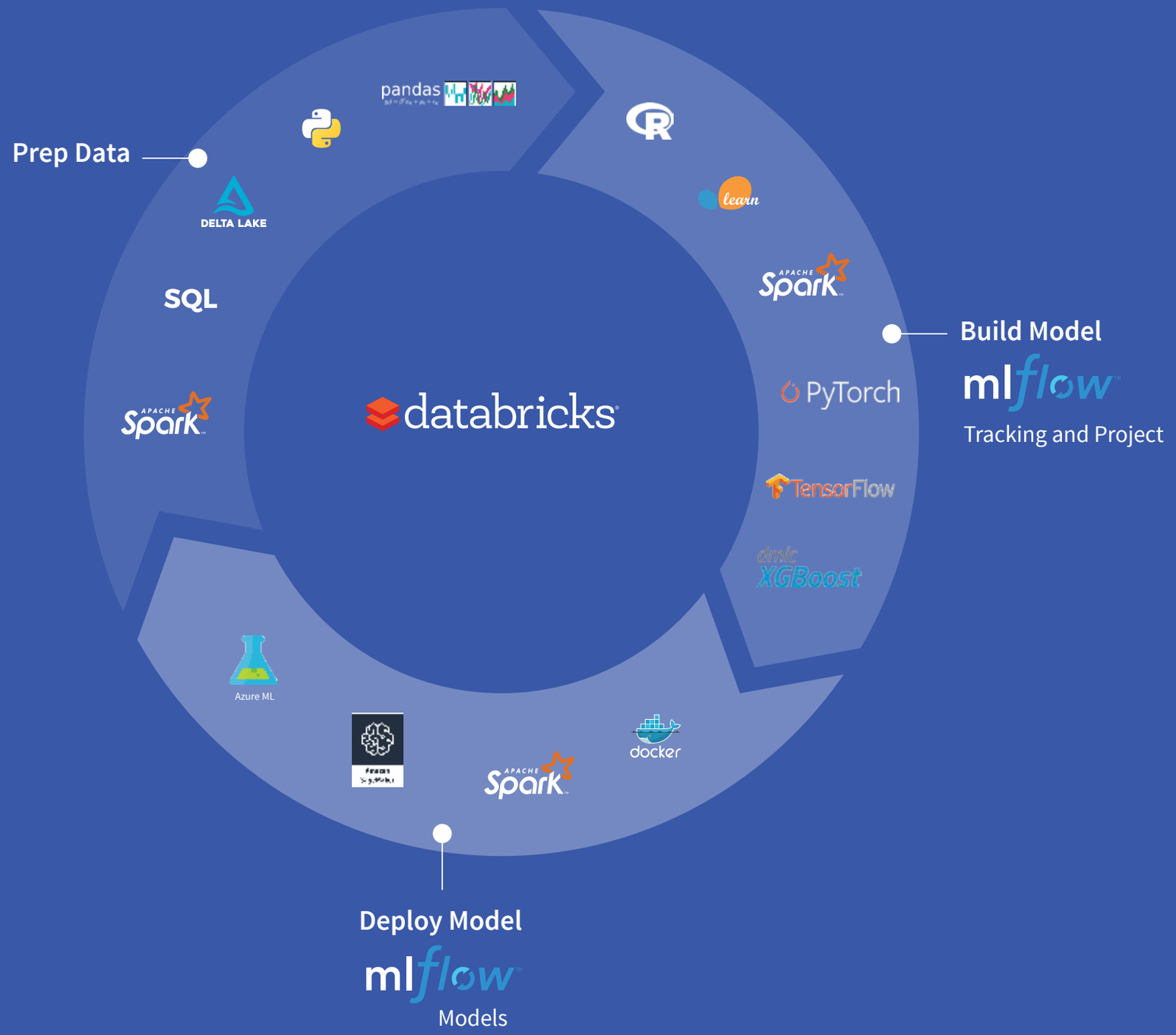
Getting to Results Faster with the Databricks Runtime for ML

Databricks Runtime for Machine Learning provides data scientists and ML practitioners with on-demand access to ready-to-use Machine Learning clusters that are pre-configured 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 on-click access to ML clusters, optimized performance on popular ML algorithms, and simplified distributed deep learning on Horovod and GPUs.



Standardizing the Machine Learning lifecycle on Databricks



Getting Started

Get started with a free trial of Databricks and start standardizing your ML lifecycle today

databricks.com/try

Contact us for a personalized demo

databricks.com/contact

Learn more

databricks.com/mlflow

Join the community

mlflow.org

Comparison Matrix

	Open Source MLflow	Managed MLflow on Databricks
Experiments Tracking		
MLflow Tracking API	✓	✓
MLflow Tracking Server	✓ Self-hosted	✓ Fully Managed
Notebooks Integration	X	✓ with Databricks Workspace
Folder/file System	X	✓ with Databricks Workspace
Big Data Snapshot	X	✓ with Delta Lake
Reproducible Projects		
MLflow Project API	✓	✓
GitHub & Conda Integration	✓	✓ with Databricks Runtime for ML
Scalable Cloud/clusters for Project Runs	X	✓
Models Deployment		
MLflow Model API	✓	✓
Built-in Batch Inference	X	✓ with Databricks Runtime
Built-in Streaming Analytics	X	✓ with Delta Lake
Security and Management		
Automated Updates	X	✓
Security Model	X	✓