EBOOK

# How to Automate Your Machine Learning Pipeline

databricks

# Contents



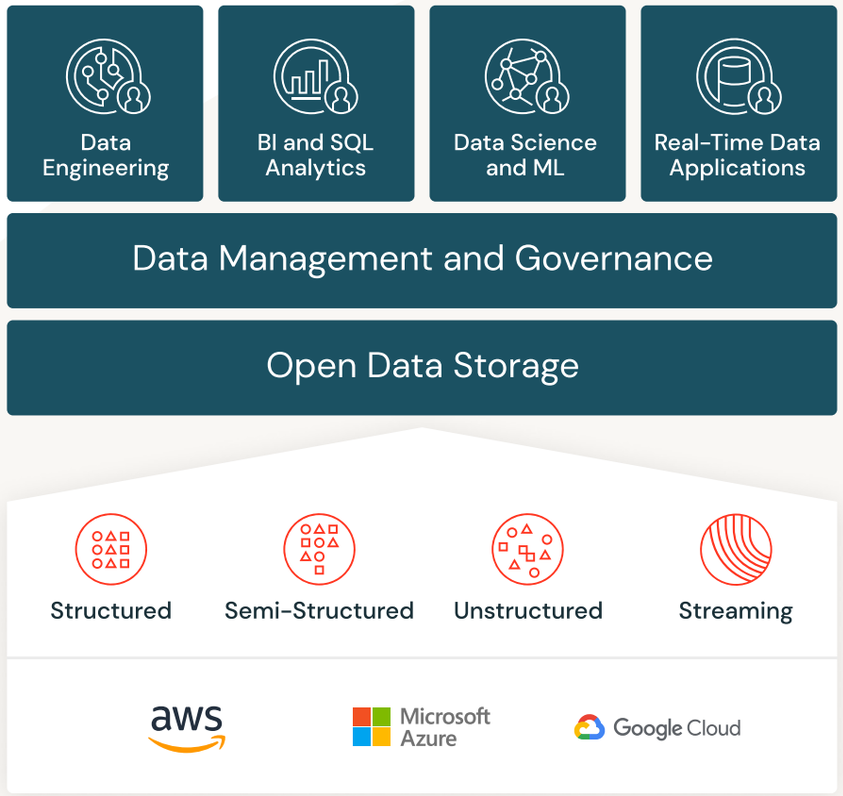
databricks

# The Data Lakehouse: Because Machine Learning Should Be Data-Native

It is counterproductive to run machine learning in a silo. The most successful machine learning projects are always built in closely collaborating teams of data engineers, data scientists and MLops engineers. Automating the machine learning lifecycle is easier with the lakehouse — a platform that is built to bring together these personas under a common operating framework, regardless of the type of data, model or business problem.

When you're choosing a data science and machine learning platform, make sure it is **data-native**. This means you can perform all your exploratory data analysis, model training and model serving in the same place you do the rest of your data management. In this paper, we will dive into how we can automate the machine learning lifecycle using the Databricks Lakehouse Platform.

A lakehouse platform can provide whatever kinds of data you want to use in your data science and machine learning initiatives. It eliminates the need to export your data to an external system or to write back to where you're managing your data. Everything stays in one place.




databricks

# Introduction

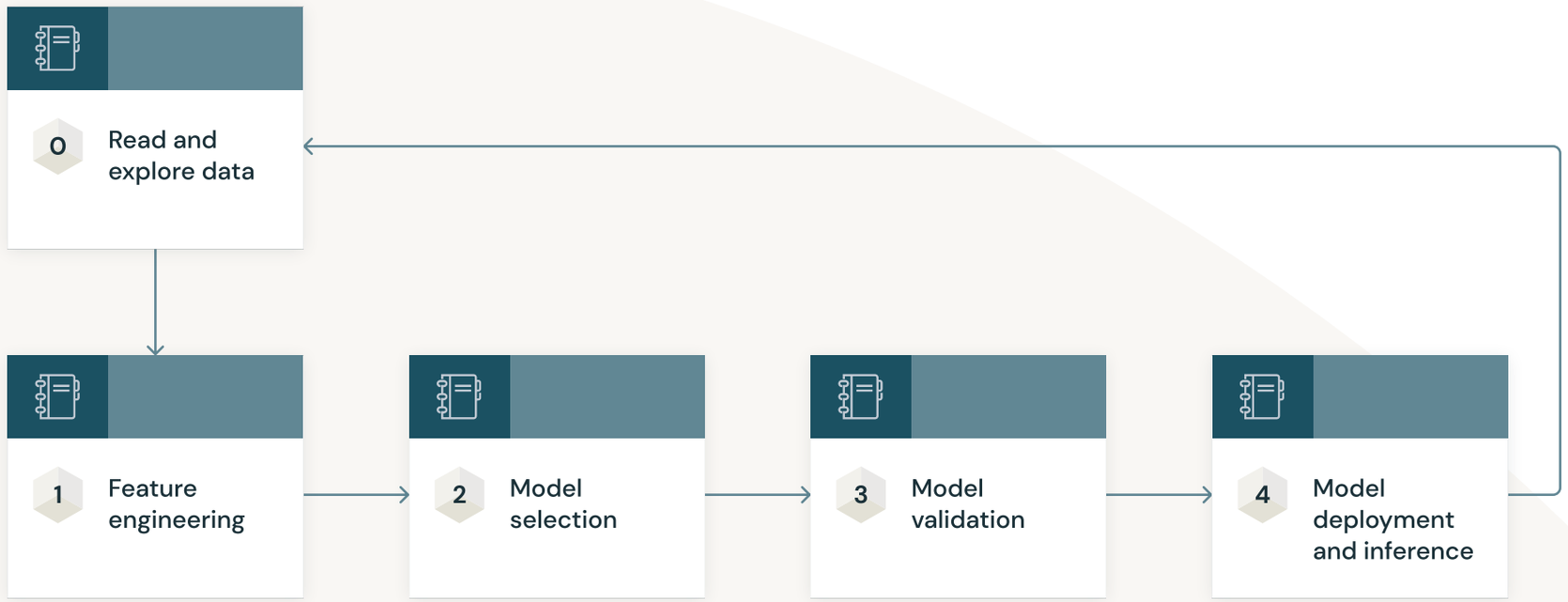
Think back to your first development cycle in data science. A line-of-business leader or your manager probably came to you with a business question for you to solve.

Your first task was to **determine which data sets were available** to you. If you were convinced that you could solve the problem with machine learning or advanced analytics, you proceeded to **feature engineering** — the process of taking raw data and transforming it into a format that could be passed to an algorithm to train a model.

Next, you moved on to **model selection**, trying lots of algorithms until you came up with the best model. In **model validation**, you took your model back to your manager to confirm that it met your business criteria and satisfied regulations. Assuming you passed these tests, you **deployed your model**, made predictions and added value to your business. Most model deployment happens in batch mode, but it's possible to run the model manually as a batch job, or as part of a streaming job.

Regardless of the deployment approach you use, the artifact that results will be a particular model object. Once you've successfully completed this process, you'll need to go back periodically to read the latest data and verify that the model still conforms to your business case.

Your first model development cycle



# Why Automate ML? And When?

What we've described here are excellent first steps — but as soon as you've completed them, you should start thinking about the consequences. How will you scale this practice as you grow? Which of these components should you automate? Just as importantly: which components should you not automate?

This may seem like a strange question to software engineers and data scientists, who are used to automating wherever possible. **Why would we ever not want to automate?** There are two reasons:

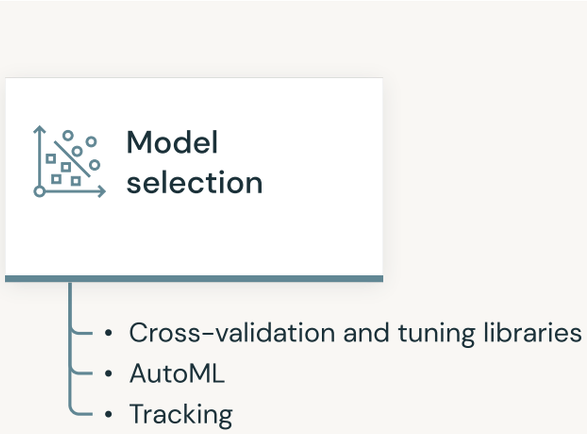
- You need to have a human involved in the process
- You want to prioritize other work over automating certain tasks

We can apply these reasons to the stages of the data science development cycle. It's unlikely anyone would be able to automate the process of determining which business questions to answer and which data sets are available to help them do it.

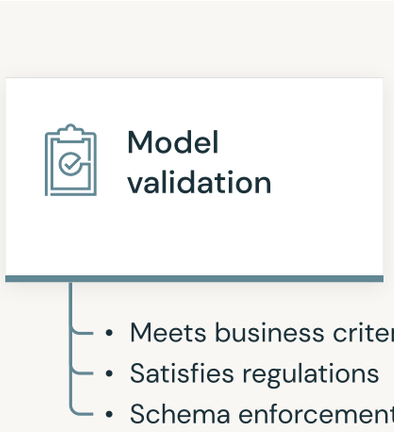
For **feature engineering**, there are open-source tools that will perform much of the automated work for generating features. These solutions will figure out which columns and variables in your data sets have a structure that would make good features:

- Featuretools is an open-source library
- Feature Factory is a project from Databricks Labs
- The Feature Store and similar solutions are gaining prominence because they enable specialized data management for machine learning. They also provide some degree of automation.

During **model selection**, you can use built-in functions for hyperparameter tuning or cross-validation. This automation helps you find the best model faster without writing code. AutoML does provide some open-source solutions, but you'll most often need to use a commercial solution. In any case, you can use automation to figure out what the best model is.




In **model validation**, it's harder to automate because you're typically dealing with external systems. Up until this point, we were simply working with our data, libraries and environment. At these later stages, we seek to communicate with the business and regulatory agencies and ensure we're satisfying their criteria. Since we're validating the model in a broader context, we're dealing with a metadata problem: any supplementary documentation or explanation needs to be traceable to a model running in production.



As for **model deployment**, we're handing off our model to another team, or placing it into a different piece of infrastructure. How can we automate this interaction with external teams and systems? Do we need a human in the loop?

Model deployment and inference

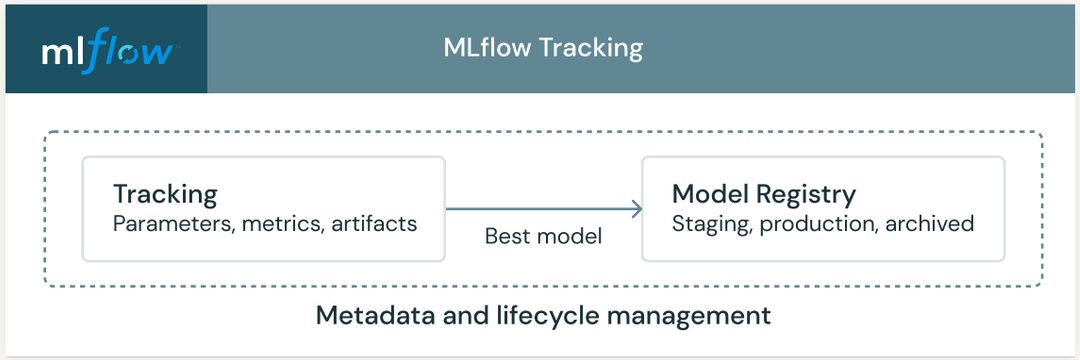
- Batch
- Streaming
- Online

To answer these questions, let's suppose our model was successful but it needs to be updated weekly. Every week, you open your laptop and manually walk through the process of ensuring that the model still meets the needs of the business. You generate a model artifact, then meet with your manager to validate it before handing it off. After six months, you've generated about 25 models. Someone asks you about your results from week 3. Because this was so far in the past, you struggle to provide insights — even if you've been tracking all your metadata in spreadsheets each week.

# How MLflow Enables Scalable Model Lifecycle Management

A better approach is to use MLflow for scalable model lifecycle management and metadata management. Here's what you'll do:

1. Use MLflow during model training to track all the parameters, metrics and artifacts you are generating.
2. Take the best model and commit it to the model registry.
3. Deploy the model as needed for inference.
4. Manage the model in MLflow for its entire lifecycle.




How does MLflow work? We can best illustrate with an example. Suppose you're using a data set to try to determine which customers are most likely to churn. You'll split data into training and test sets, and you'll use it with scikit-learn. MLflow can help you collect the metadata around your model automatically. Here's the procedure:

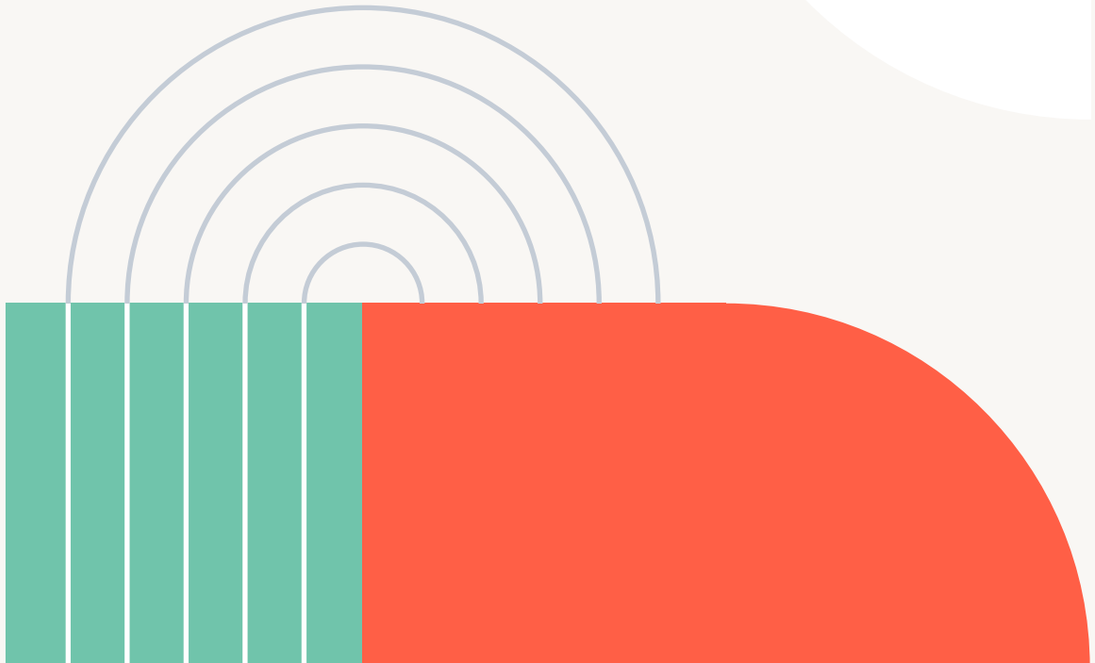
1. Set the experiment where all the runs will be logged.
2. Perform a training run to fit the model, using scikit-learn.
3. Collect metrics for the training run.
4. Log the parameters and metrics using MLflow APIs.

Your Databricks notebook will log the results of this training run. You can use APIs to wrap all your accuracy checks and parameters around your code so that every time you train a model, the results will be logged to MLflow for further exploration. But there's an even easier way to do this.

With Databricks auto-logging, you don't need to write any code to fit your model. You can simply use MLflow, scikit-learn and auto-log to run the same model-fitting code. You'll generate far more information about your model than you would have been able to do coding by hand, due in part to the decision tree classifier in scikit-learn. All of this happens when you turn on auto-logging in your Databricks environment. Meanwhile, you can easily review all your test runs in an intuitive experiment view and add notes and tags.

As you use MLflow, you benefit from integration with Apache Spark™, which was created by the founders of Databricks. By accessing this massively scalable parallel processing engine, MLflow enables you to orchestrate the training of many models simultaneously and log all their results to the tracking server. HyperOpt, a Python library that uses Bayesian optimization to determine the best parameters for a particular model, also works seamlessly with MLflow. As a result, you can parallelize your tasks on a massive scale, log all results and easily converge upon the best model.

As metrics and parameters come in from each run, MLflow automatically generates fully editable notebooks for you. You can see exactly which types of preprocessors were used and how columns were standardized. You get detailed information on feature importance as well as how to use each model for inference. And because MLflow is built on open-source technology, you can recreate any of its results as needed.



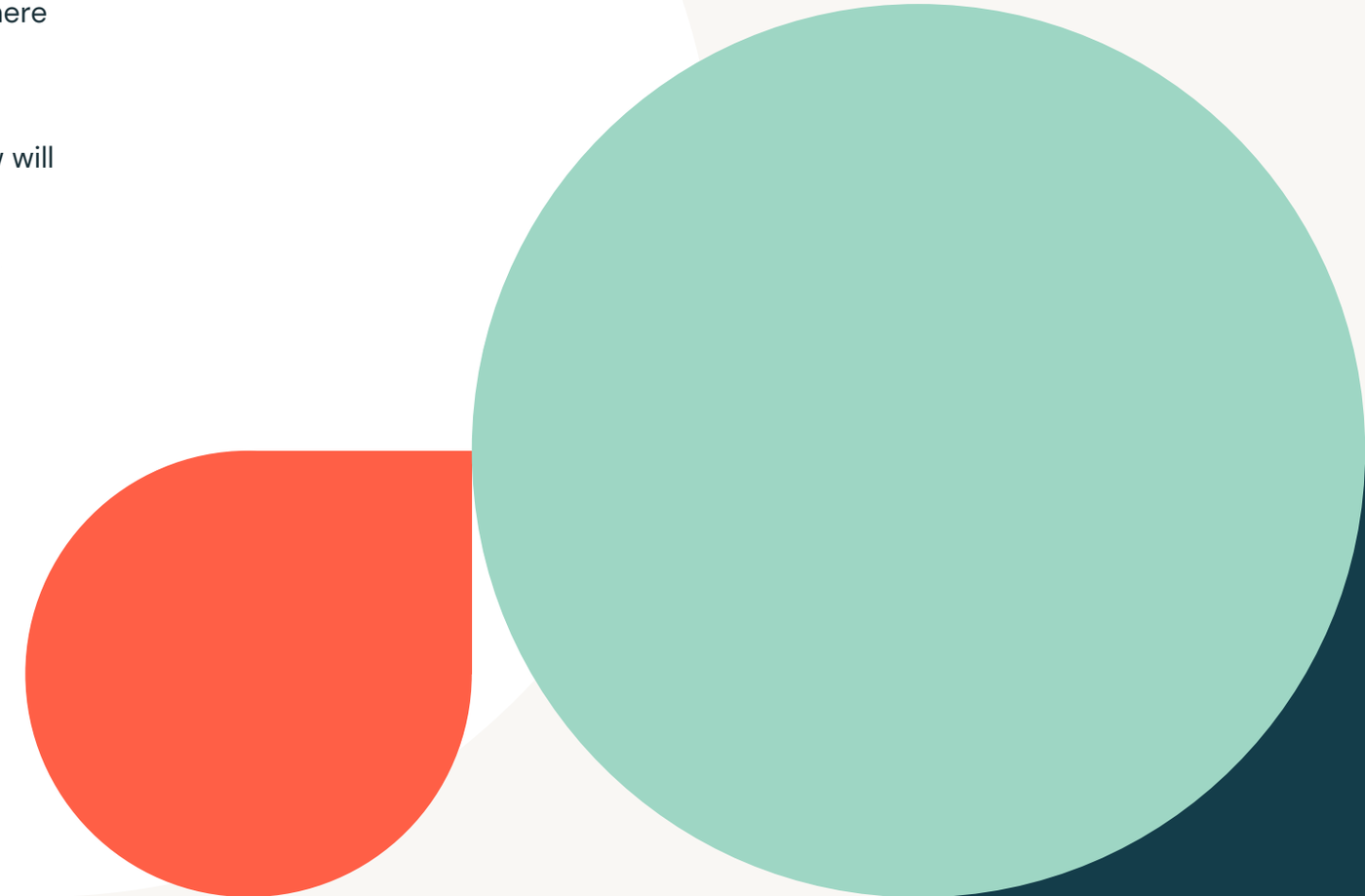
databricks

# Streamlining Model Validation

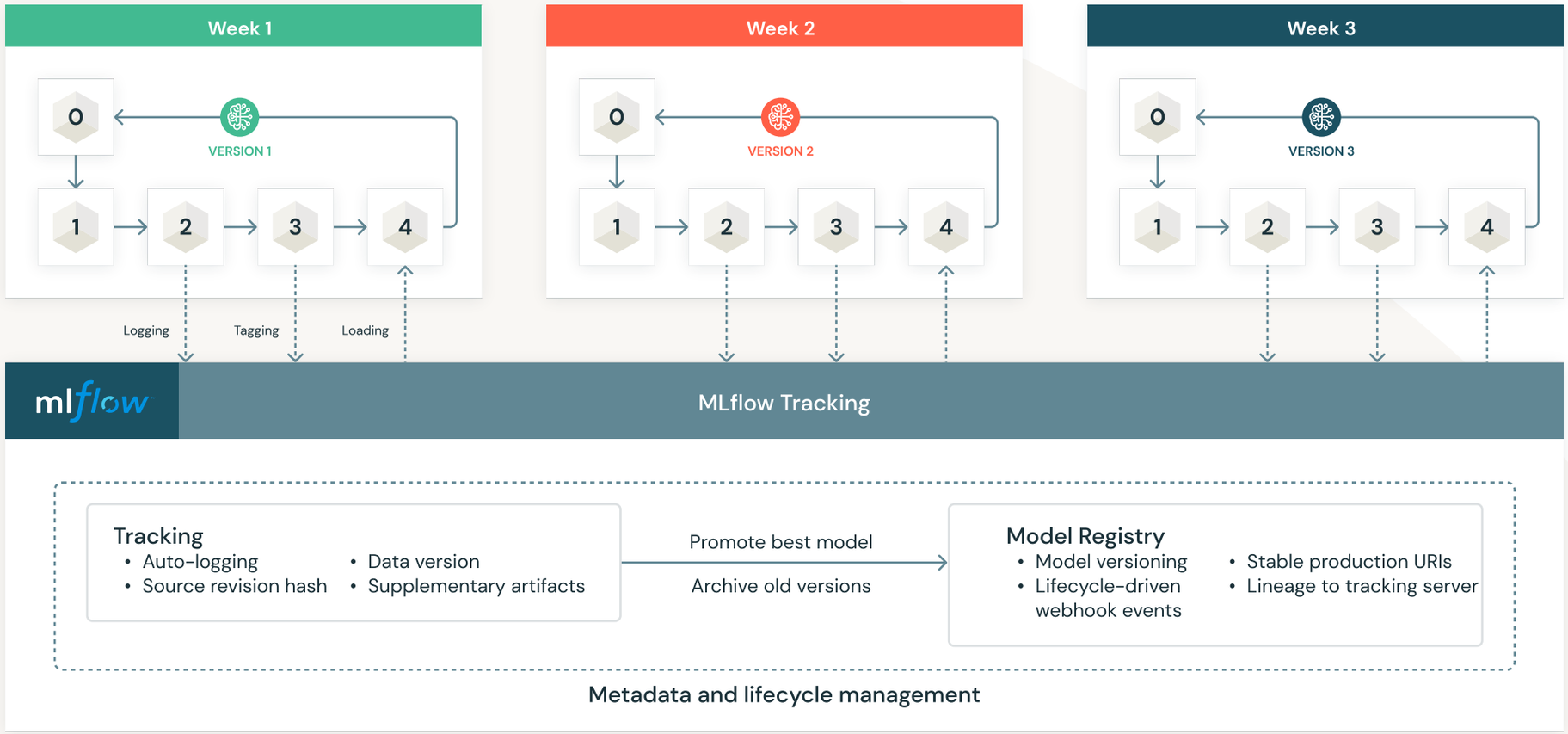
Once you've identified the best model and want to move it to the registry, you'll need to validate that it meets business criteria. MLflow can help you introduce automation around this critical step using webhooks — an event-driven API call.

Suppose a member of your team promotes a model to your registry. Based on this event, webhooks can send a Slack message, call a cloud function in AWS or Azure, or run a Databricks job to start the model validation process. Here's a scenario:

1. A data scientist uses a notebook to promote a model to the registry.
2. The model will move to the registry but be marked as "none" or "no stage."
3. The movement of the model will trigger two webhooks to fire. One will send a Slack message that notifies the team that the data scientist wants to move the model to staging. The other will be a call to the Databricks jobs API, where the model will be run in a separate testing notebook.
4. The model will be tested for adherence to regulatory criteria, proper documentation and accurate performance on subsets of the data. MLflow will automatically tag the model with the results of testing.
5. If the model passes, it will be approved and move into staging. If it doesn't pass, it will be archived, and the data scientist will be notified.



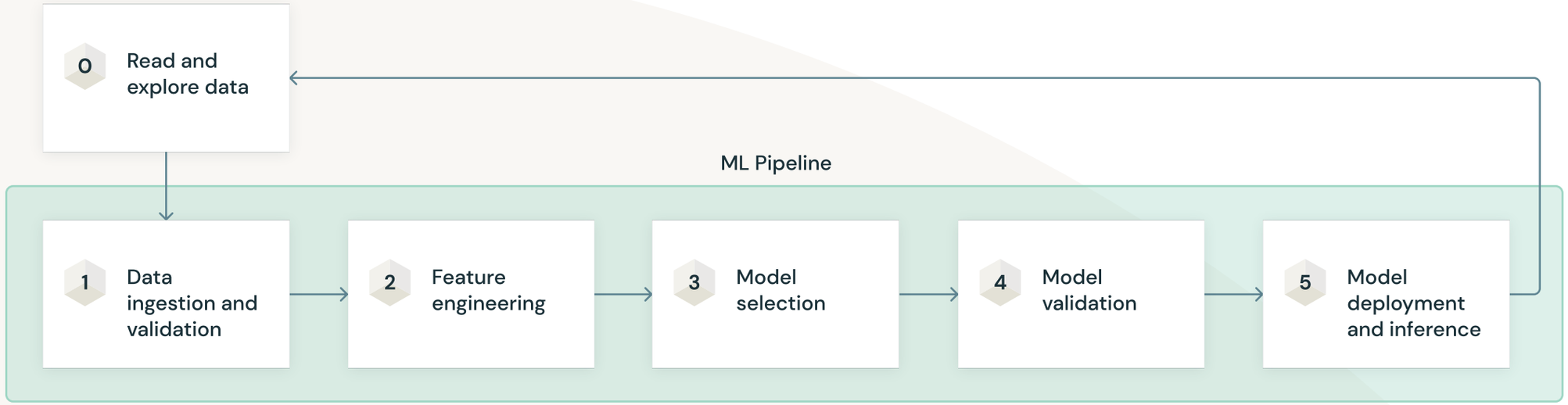
## Automated model metadata and lifecycle management with MLflow on Databricks




Once models are in the registry, you can load them into memory and perform inference with batch or streaming jobs, as well as online inference via REST API. You can also sunset old versions to ensure you're always using the most recent models.

# Automating Your Entire ML Pipeline

So far, we've explored how MLflow can automate various components within your ML workflow as you develop, track and deploy model artifacts. What would it take to automate the workflow itself?



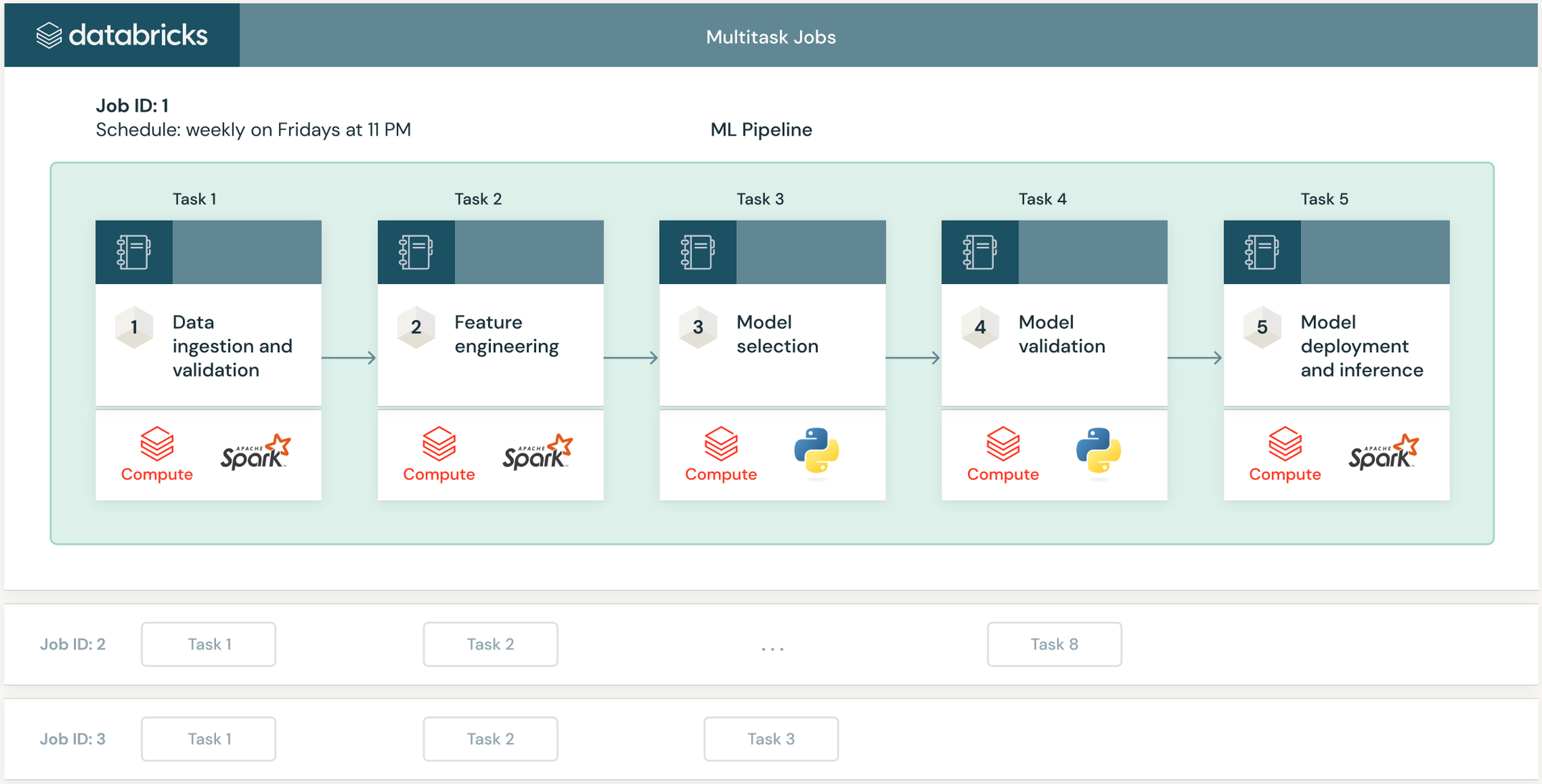

MLflow treats your entire ML pipeline as a process to be automated and modularized. When you orchestrate each of your ML tasks to run as a unit, you can mature your practice and create a whole that's greater than the sum of its parts.

MLflow sees the pipeline as a multitask job. It all begins when you choose a notebook or task to run. You may also specify dependent libraries or set up notifications. Once Databricks has performed data ingestion and validation, it moves directly into feature engineering. You can automate model selection with the Databricks AutoML API. When the chosen model moves to the registry, it will kick off the validation and testing process. Once a model passes testing, it will be deployed. The system will then run a batch job to score new data.

All of this happens without human intervention. You can run the entire workflow on a schedule — such as every Friday evening at 11:00 PM — or on an ad hoc or event-driven basis. You retain the flexibility to allocate different resources to each task. Although the tasks are all part of a single workflow, they're isolated in terms of resources and environment, which enables you to ensure adequate computing power and memory at each stage.

**For general job orchestration:**

- Resource and environment isolation
- Linear and nonlinear flows
- Retries, alerting

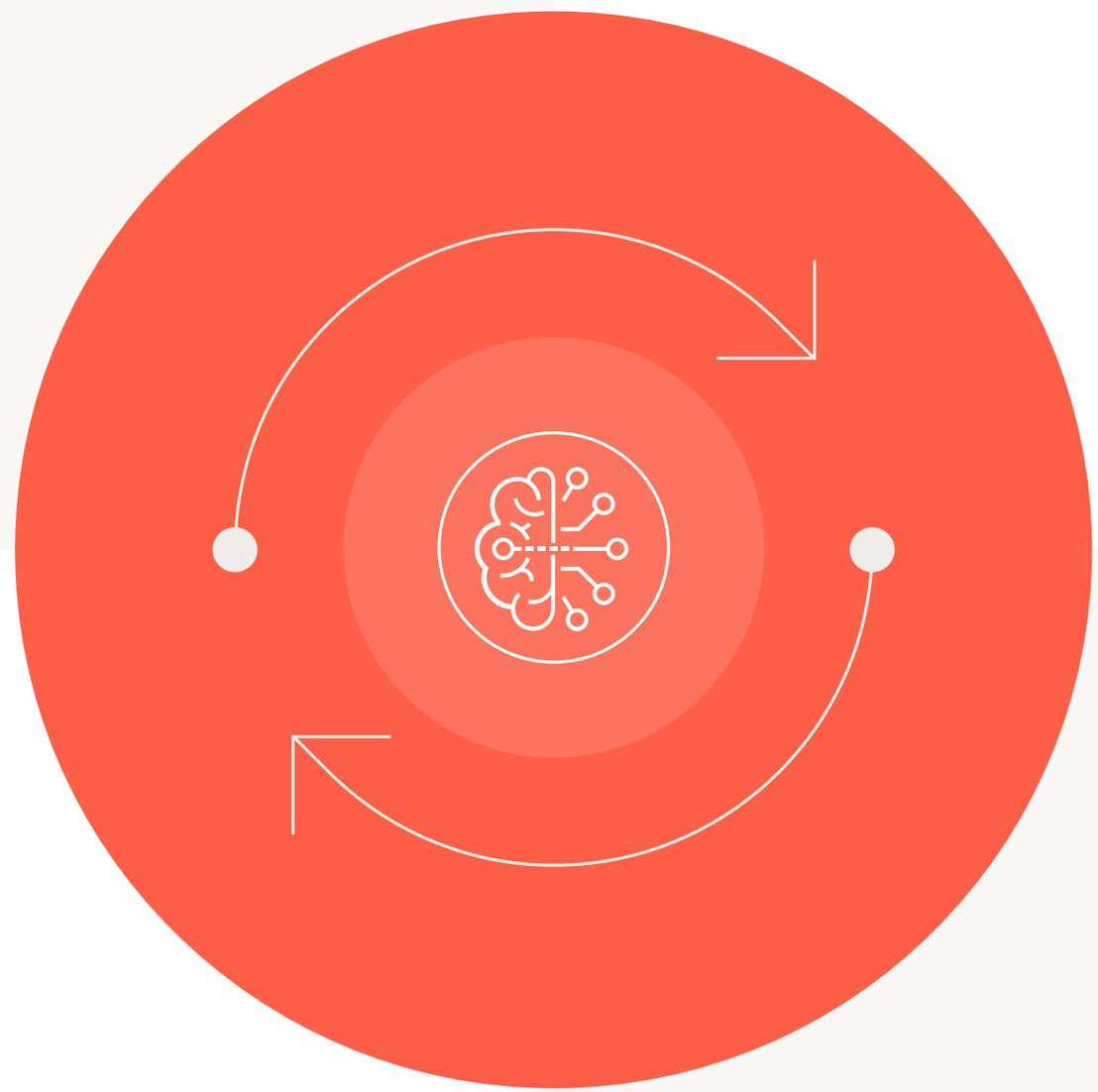


# Incorporating Continuous Integration and Continuous Delivery

As they explore the possibilities of ML pipeline automation, many data science professionals wonder how they can incorporate CI/CD practices. Databricks Repos is designed to work with whatever version control system and processes you already have in place — including Jenkins, Azure DevOps and other solutions. You can integrate with these systems and build off your current processes.

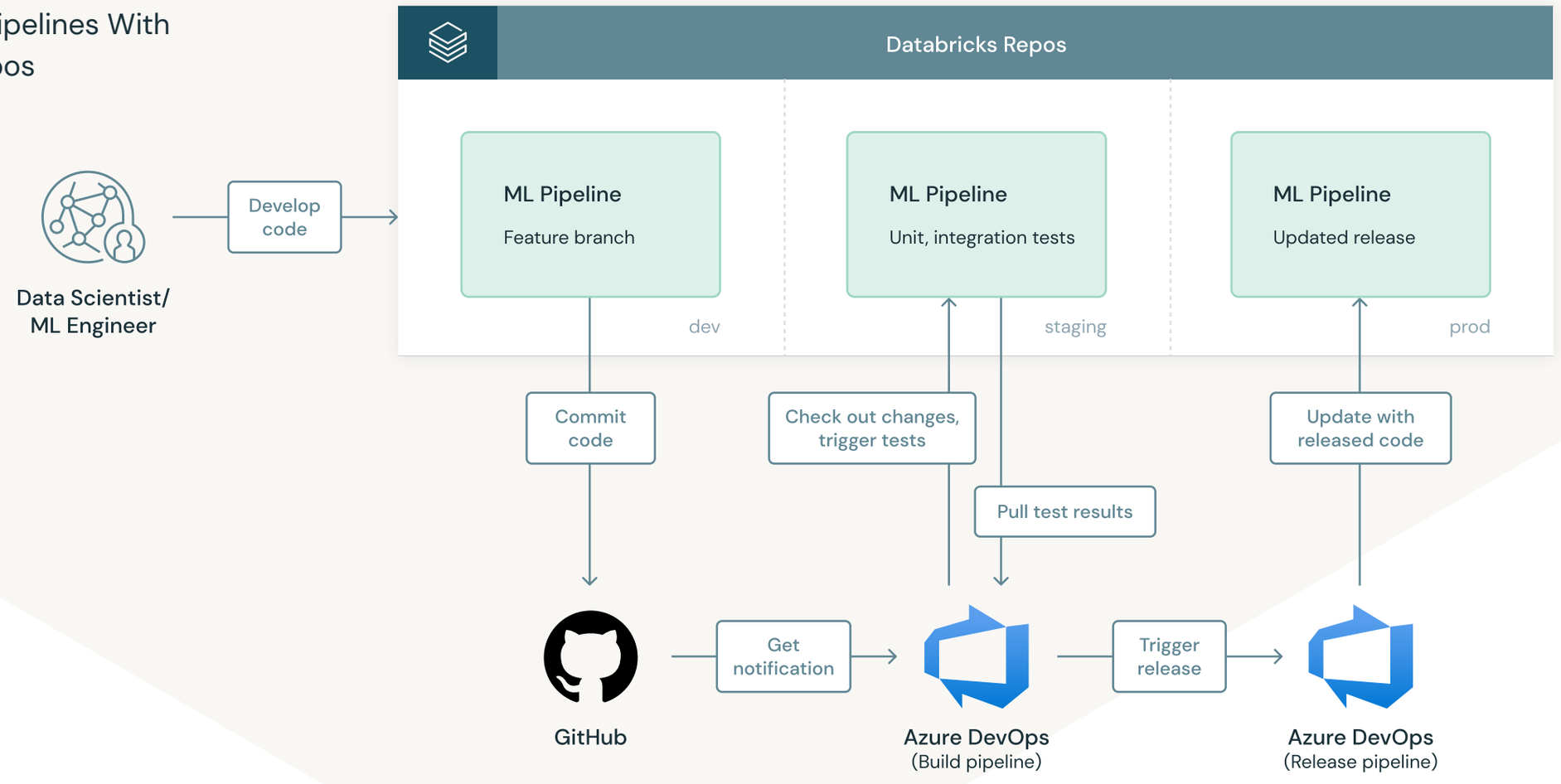
- For **continuous integration**, you can deliver new code and new features into the pipeline if you're updating a particular task
- For **continuous delivery**, you can perform a full test of the entire pipeline in an environment that mimics the production environment before you update the pipeline in production

Suppose you're a data scientist working in Azure DevOps. All your checked-in notebooks will be synced with a GitHub repo with Databricks Repos and will be visible in the workspace. When you develop a feature branch and commit your code to GitHub, Azure DevOps will get a notification from GitHub, which will cause it to check out the changes. You can then run tests in Databricks in an environment that mirrors your production environment.



databricks

## CI/CD for ML Pipelines With Databricks Repos




Once all the tests are done, the code will be pulled back to Azure DevOps and trigger a release. The final pipeline will run to update the code that's running in production.

**Source:** Part 1: Implementing CI/CD on Databricks Using Databricks Notebooks and Azure DevOps
**Credit:** Alex Ott, Michael Shtelma, Piotr Majer

databricks

# What to Remember as You Get Started

Automating your ML lifecycle is simpler than it seems. Just follow these concepts:

Automate the tedious things

Don't forget to use human judgment wherever it's needed

Simplify development and set yourself up for scalability with machine learning on Databricks


Use MLflow to keep everyone accountable for the models they're building and to scale out to many use cases

Use AutoML to bootstrap particular use cases

Databricks provides powerful automation capabilities in a glass-box format so that your team can easily review and modify individual tasks or the whole workflow. Whether you're a beginner or a seasoned ML professional, you can lean on Databricks to orchestrate your entire ML pipeline.

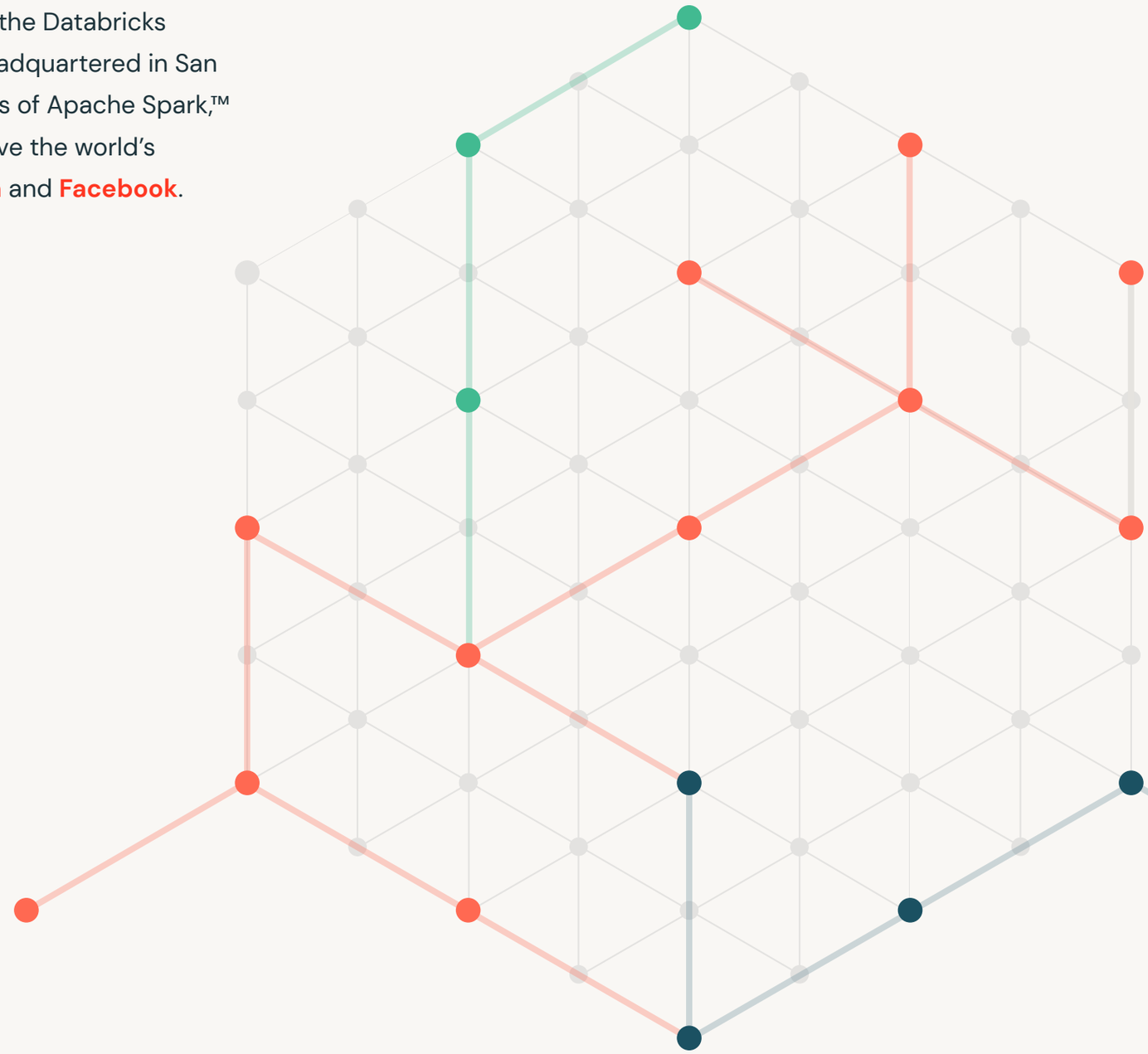
Get started now. Request a **14-day free trial of Databricks**.



databricks

# About Databricks

Databricks is the data and AI company. More than 5,000 organizations worldwide — including Comcast, Condé Nast, H&M and over 40% of the Fortune 500 — rely on the Databricks Lakehouse Platform to unify their data, analytics and AI. Databricks is headquartered in San Francisco, with offices around the globe. Founded by the original creators of Apache Spark,™ Delta Lake and MLflow, Databricks is on a mission to help data teams solve the world's toughest problems. To learn more, follow Databricks on **Twitter**, **LinkedIn** and **Facebook**.



Get started with a free trial of Databricks and start building data applications today

START YOUR FREE TRIAL

databricks