Brought to you by

databricks

MLOps

for dummies®
A Wiley Brand

Databricks Special Edition

Scale your ML operations efficiently

Harness the opportunities of machine learning

Encourage collaboration and reduce risk

Steve Kaelble

About Databricks

Databricks is the data and AI company. Thousands of organizations worldwide — including Comcast, Condé Nast, Nationwide, and H&M — rely on Databricks' open and unified platform for data engineering, machine learning, and analytics. Databricks is venture-backed and 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 social media:


twitter.com/databricks


www.linkedin.com/company/databricks


www.facebook.com/databricksinc

MLOps

Databricks Special Edition

by Steve Kaelble

for dummies®
A Wiley Brand

MLOps For Dummies®, Databricks Special Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN: 978-1-119-98167-1 (pbk); ISBN: 978-1-119-98168-8 (ebk). Some blank pages in the print version may not be included in the ePDF version.

Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

Project Manager: Carrie Burchfield-Leighton

Sr. Managing Editor: Rev Mengle

Managing Editor: Camille Graves

Acquisitions Editor: Ashley Coffey

Business Development Representative: Matt Cox

Table of Contents

Introduction

The adoption of machine learning (ML) has grown phenomenally in the past decade. It wasn't all that long ago that only the biggest technology companies could afford all of the resources required to make ML happen. But as with most great technologies, this one is becoming more and more ubiquitous — across all industries, serving organizations of all sizes.

That doesn't mean developing and deploying ML models is child's play. Make no mistake, it's incredibly complicated (and speaking of mistakes, you definitely don't want to make any). As ML makes its benefits known to business stakeholders, they want it more and more. So, how can the data scientists and ML engineers turn this miracle of technology into more of a mass-produced opportunity?

That's what this book is all about: a concept known as *machine learning operations* (MLOps). Like all the technology approaches that end in "Ops," this one is about operationalizing the complicated stuff, gaining long-term efficiency, reducing cost, and improving stability. As long as I'm referring to those other "Ops" approaches to technology development, let me just go ahead and put it this way: MLOps = ModelOps + DataOps + DevOps.

About This Book

MLOps For Dummies, Databricks Special Edition, is your guide to this improved way of approaching your ML life cycle management. Read on and you discover how MLOps helps you truly scale your operation and manage dozens or hundreds of ML solutions simultaneously. This book shows you how MLOps helps your operation reduce waste, encourage collaboration among your various participants and stakeholders, and improve implementation. It outlines the components of MLOps and details how the process works, from exploratory data analysis to feature engineering to model training to monitoring.

MLOps is a general concept that can work with any set of tools and applications, but in this book, I use examples that tap into Databricks' features and functionality.

Foolish Assumptions

In writing this book, I make some assumptions about you:

- You're an ML practitioner of some sort, or perhaps you aren't, but you manage a business area that benefits from ML.
- You may have decent knowledge about the concepts of ML and training models, or you may be open to learning more about ML and how it can help your business area.
- You'd appreciate some ideas about how this work can happen at a greater scale with more efficiency.

Icons Used in This Book

In the margins of this book, you see some clever little icons. They look nice, but they're more than just a pretty face.


REMEMBER

While I've carefully crafted several thousand words on the topic of MLOps, if you're short on time, at least don't miss the paragraphs marked by the Remember icon.


TIP

Tips represent actionable information. That's what you're after here, right? These icons spotlight a helpful hint for MLOps success.


WARNING

Anything technology related can have horrifying potential downsides. The Warning icon points to something you need to really keep in mind to avoid those pitfalls.

Beyond the Book

This book has information on MLOps, but it's only the tip of the iceberg. If you come away from your reading with an appetite for more information, here are two places to turn:

- `databricks.com`: Get full information about the Data Lakehouse platform that helps enable MLOps.
- `mlflow.org`: This site provides an open-source platform for the ML life cycle.

IN THIS CHAPTER

- Exploring the benefits of AI
- Recognizing where ML work can go awry

Chapter 1
Understanding the Current ML Landscape

Artificial intelligence (AI) and machine learning (ML) have gained acceptance by leaps and bounds. Enterprises are finding new tasks they can automate every day, not just in predictive analysis but in daily operations. So, in this chapter, you explore the expanding use cases for these technologies. I also go into detail about the challenges of creating ML solutions — there are many ways that smart people with good intentions can end up heading down the wrong path.

Expanding the Power of AI

For many years, AI has made for some great science fiction — including a 2001 Spielberg movie by that very name, *A.I. Artificial Intelligence*. It's a dramatic thing to ponder, the thought of machines able to think for themselves, and in the case of sci-fi, that drama has a tendency to turn dystopian.

In real life, meanwhile, people in recent years have been captivated by futuristic applications of AI, especially use cases that could have a powerful bearing on daily tasks. Self-driving vehicles,

for example. The current landscape, though, is marked by ongoing developments that may be less noticeable to the average person, but no less dramatic in implication.

The big trend driving the AI and ML story is the power of data. The most successful companies tap more and more into the potential of their data. That has often meant accessing data stored in relational databases, performing analyses, and gaining insights. But the current landscape is taking the power of data much further.

REMEMBER

These days, data isn't just driving analyses; it's also becoming more involved in operational processes. Enterprises are becoming increasingly automated with the help of powerful capabilities of ML, which is essentially a subset of AI. ML is the application of predictive models and data science, and that's the focus of this book.

Customer support, for example, can increasingly be delivered by ML-powered chatbots, which solves hiring challenges and frees up humans from some of the more mundane customer-service tasks. These kinds of technologies are processing mortgage applications. They're spotting cyberthreats more effectively than people. They're helping supply chains automatically adapt to the ballooning challenges of recent years.

In short, organizations aren't just using what they used to call "Big Data" to inform their big decisions. They're incorporating that data into their operations, creating new ML applications, giving those applications new responsibilities, and achieving amazing things. These things are often happening behind the scenes in ways less obvious and head-turning than, say, robot children or cars with no drivers — but the advances are at least as revolutionary.

You can trace a lot of these advances to the proliferation of cloud data warehouses and lakehouses. These technologies solve the challenge of not just storing massive amounts of data, but making it useful in ways that are more affordable and less complex. Organizations have been wrapping their arms around Big Data for a decade or more, but their ability to store and effectively process that data has never been better.

REMEMBER

The current landscape is boosting enterprises' capabilities in both business intelligence and ML. They're gaining the ability to dump massive amounts of data, both structured and unstructured, into central repositories that they can access for a wide range of purposes. Their data is accessible for historical analysis as well as predictive functions driven by ML and data science.

Uncovering the Gaps in ML

There's no doubt that ML can be downright magical. It can answer difficult problems and make astonishingly accurate predictions. It can tackle massive tasks, handle monotonous work without complaint, and make a lot of money for companies that know how to tap into it. But it's not exactly child's play. It's complicated, it's hard, and you can take plenty of wrong turns if you're not careful. The discipline of ML engineering is all about navigating this complexity. This section covers some thoughts about the problems that are in need of solutions.

Planning problems

WARNING

The biggest cause of failure is neglecting to plan projects thoroughly. It also is one of the most embarrassing and demoralizing causes of failure, because it *can* be avoided.

Here's a specific example that helps illustrate how careful planning can lead to greater success. Imagine that the marketing team wants to tap into ML to improve how they use email communications for letting customers know about upcoming sales they might find tempting. The marketing executive simply says, "make those click and open rates go up."

The problem is, many variables could address that too-general challenge. Does the marketing team want to be able to craft customized emails? Get ML predictions for attaching more relevant subject lines? Anticipate which products to put in front of the customer?

More planning is needed, along with more information from the stakeholders. What if it turns out the marketing executive just needed to know what time of day is best for sending emails to particular customers? Being aware of that specific need is essential

planning information — without it, the data science team may have gone down all those other paths that all have some merit but have no chance of meeting the actual requirements of the marketing team.

TIP

To avoid potential planning problems, the following list contains important planning questions to ask:

- What is the project?
- Is there a current process, and if so, what is it?
- What business need is being solved?
- How do you need to use the information generated?
- What is the timeline, including when it will need to be tested?
- What outcome would you consider to be a success?

As you can see, planning focuses on the problem to be solved. If you dive too quickly into talking about implementation details, you may miss out on a full understanding of the problem. And that's a problem.

Scoping problems

When it comes to project scoping and research, internal stakeholders will have two big questions regarding an ML project:

- Will this project solve my problem?
- How long will it take to be up and running?

Sounds simple enough. But the reality is, scoping an ML project is incredibly challenging because you can approach any particular stated goal in a number of different ways.

Imagine that you have a couple of different data science teams, both of which are presented with a need to develop a solution for the increasing problem of fraud in the billing system. One team has mostly junior data scientists, who've spent their careers in the data science workforce without a lot of time in academia. The second team is filled with Ph.D. academic researchers.

The first team will search for similar solutions that others have already developed, scouring blog posts and open-source data examples. There may be many such articles available, and the

team may feel like it'll take a couple of weeks to pull in some existing code and come up with an XGBoost binary classification model. Spoiler alert: This team may be underthinking the problem and solution, and it may end up failing.

The second team will look instead through published research papers on the topic of fraud modeling. They'll spend days digging through journals and papers, arriving at some cutting-edge ideas on detecting fraudulent activity. They suggest it'll take several months to implement a solution, including training and evaluating a deep learning structure. Spoiler alert: This team may fail, too, but for different reasons.

In the first case, the problem may well be a lot more complex than the examples in some blog post. The junior data scientists run the risk of coming up with a model with unacceptable false positive and false negative rates. In the second case, the solution may be highly accurate and pretty cool, but odds are slim that the team would be granted enough resources to dive into a risky, cutting-edge concept to solve a business problem.


TIP

The lesson is that you need a balance of both ways of thinking. The academics are important, for sure, to help you advance your knowledge and research into algorithms. But you also need application-oriented ML engineers with a laser focus on the business problem at hand. A middle ground is your best bet for scoping and research, to minimize the chances of estimations that are wildly wrong or unrealistic, one way or another.

Development issues


WARNING

A poor development practice for ML projects can come back to haunt the team as the project progresses. It's not as dramatic a problem as some of the other glitches, but the bottom line is that if you have poor development practices and a fragile or poorly designed code base, your project work gets harder, it's more likely something will break in production, and it'll be more difficult to make improvements later on.

A good example is working with a monolithic code base as opposed to a modular one. Each developer who wants to make specific changes has to do a lot of scrolling and searching to find the right sections to alter. And if multiple developers all have

different things they need to edit, they can easily run into version control problems and the need to figure out which copy is the "master" copy.

A modular approach, on the other hand, involves working on code base in an integrated development environment — with the code base fully modularized and registered in Git. Each different developer can check out a feature branch from the master and make small edits, and even do so simultaneously.


TIP

The modular approach is a big change from the approach many data scientists have used, with interactive notebooks for prototyping ideas, experimentation, and analysis. But it's a much better way of doing business, simplifying work, allowing simultaneous development of different sections, and offering the ability to test new functionality easily to eliminate pesky, hard-to-track bugs.

Experimentation issues


WARNING

When you hit the experimentation phase, a couple of common things can go wrong. Your work can go off the deep end with complexity, testing too many things or fine-tuning an approach for too long. Or, the prototype may be underdeveloped and end up performing so badly that the organization gives up on it and moves on to something else.

The first scenario can happen when a team of research purists dives in with not enough grasp of real-world limitations. They may spend weeks exploring cutting-edge research papers, arrive at multiple broad potential solutions, and set out to test each one thoroughly.

It's not that any of the experimentation is necessarily misguided — the problem is that the approaches needed to be narrowed down in the development stage of the project. If a team takes on too much work in the experimentation phase, the result is analysis paralysis.

The latter situation could result from inadequate research that leads the team to test just a single approach. It may look good with some initial testing on cherry-picked samples, but when the model is trained on a full data set, it fails to work properly. A too-simplistic test can lead to disastrous results.

Deployment issues

In the ML world, it's possible for solutions to be almost too good. The sky is the limit when it comes to the kinds of models you can build and sophisticated predictions your models can deliver. In some cases, the real hang-up comes when you have to deploy those solutions in the real world.

Take as an example a project from a company offering analysis services to the fast-food industry. A successful model has been delivering great predictions for inventory management at a regional level, with large-batch predictions of daily demands and customer counts, delivered to each region once a week, requiring regional model retraining once a month. So far, so good.

Now imagine that the company wants a new and improved prediction system that offers inventory forecasting at the store level, rather than regionally. And predictions need to respond in near-real-time, rather than weekly predictions.

It's a complicated but doable challenge, but the serving component of the solution is a problem. It needs a REST API to serve data to individual store owners via an application, and per-store forecasts must be updated multiple times a day. The architecture becomes far more complicated, and the cost of implementing in the cloud is far higher. So much higher, in fact, that this new model costs more than the revenue it generates.


TIP

The moral of the story is, it's essential to think ahead and think carefully about deployment and serving. You've got to have a keen eye for how much it's going to cost to run, maintain, and monitor a solution. Just because it can be developed doesn't guarantee that it can be deployed cost-effectively.

This points to the virtues of simplistic architecture. With ML, you should always aim for the simplest design you can. If you need an inference once a week, a batch process should suffice, as opposed to real-time streaming. If you're dealing with data volumes measured in megabytes, go with a database and simple Virtual Machine (VM) rather than a 25-node Apache Spark (TM) cluster.

Unnecessary complexity raises the chances that something will break. And it's bound to be more costly than necessary.

Evaluation issues

Like so many other parts of the business operation, an ML project needs to be able to prove its worth. And it needs to be able to do so not just in its early stages, but once it's deployed.

Here's a sad scenario to ponder. Imagine that your team has been working on an ML initiative to increase sales through predictive modeling. You've done everything right, followed best practices, hit deadlines, ascertained that the solution does exactly what the business is seeking, ensured that you've developed maintainable and extensible code. The post hoc analysis of the predictions compared with actual results has found them to be incredibly accurate.

Now, you're in production, and have been for a few months. An executive takes a look at the quarterly cost, has a bout of sticker shock, and asks how much money this thing is making. Thus comes the question, can this ML model justify its existence?

Well, you can't just attribute all sales to the model — that would not be believable. You can't really compare this quarter's sales to the same quarter a year earlier, because there were plenty of other factors that also might have impacted sales.

What you need is A/B testing, using sound statistical methods to show how much of the growth in sales can be attributed to the ML model. But you can't really do that after you've already fully deployed the model to all customers — you needed to plan for that earlier.


TIP

The lesson is, plan ahead for how you can objectively evaluate your model's performance. You need to be able to prove its value, or else be faced with the gut-wrenching prospect of having to shut down an otherwise fantastic piece of work due to budgetary concerns.

IN THIS CHAPTER

- Understanding the MLOps concept
- Seeing the similarities to DevOps
- Outlining the guiding principles

Chapter 2
Introducing MLOps

Chapter 1 includes a sobering look at the hurdles your teams may encounter as they work to create machine learning (ML) solutions to your enterprise's needs. But never fear, because the whole point of this book isn't to bum you out — it's to offer solutions.

This chapter introduces the concept of MLOps, a sensible approach to ML engineering that helps overcome the potential bummers we've established. Read on to learn the basics and explore the commonalities between MLOps and DevOps.

Getting to Know MLOps Basics

ML operations (MLOps) is a core function of ML engineering. Like the concept of development operations (DevOps), it's all about streamlining complicated processes so that concept can move toward reality as smoothly and quickly as possible.

REMEMBER

MLOps focuses in particular on streamlining the process of bringing ML models and data into production, and after they're in production, maintaining them and monitoring them. It's a collaboration of many different players, including data scientists, DevOps engineers, and your IT professionals.

Especially to the point, the practices described under the term MLOps are intended as solutions to the problems that can arise through ML development and deployment. Or better yet, they can serve as preventive measures.

It is a fantastic concept for creating ML solutions, ensuring quality and success. The MLOps approach also helps data sciences and ML engineers collaborate and increase the pace of model development and production. Through MLOps, you'll be implementing continuous integration and deployment (CI/CD) practices, and you'll be moving down the ML path with all of the proper monitoring, validation, and governance of models.

That's no small accomplishment, either. ML isn't a piece of cake, and productionizing it is difficult. You're talking about a life cycle that runs through such complexities as data ingest, data prep, model training and deployment and monitoring, explainability, and a whole lot more.

REMEMBER

This process requires lots of collaboration, and that means many handoffs from one team to another, including data engineering and data science and ML engineering. The MLOps process includes experimentation, iteration, and continuous improvement, throughout the life cycle. If you don't have the operational rigor to keep all of these processes in sync and working together, you're sunk. MLOps helps you swim in these turbulent waters.

The top three primary benefits of MLOps are

- **Efficiency:** Through MLOps, data teams become much faster at model development, the quality of their ML models increases, and they become speedier at deployment and production.
- **Scalability:** The more you rely on ML, the more you need to be thinking about scalability and management. With MLOps, you can handle thousands of models — overseeing, controlling, managing, and monitoring for continuous integration, continuous delivery, and continuous deployment. You'll enjoy reproducibility of ML pipelines, with the tightest possible collaboration across data teams. That means less conflict with DevOps and IT, and much faster release.

- **Risk reduction:** Your models and data are likely to need a lot of regulatory scrutiny, and you need to be sure they're still working well together as intended. MLOps helps with that, enabling greater transparency and faster response to those kinds of requests. You'll achieve greater compliance with your own policies, as well as the regulations that hit you from the outside.

Data scientists and software engineers use an MLOps platform to create a collaborative environment. Through that environment, the collaborative players achieve the following:

- Iterative data exploration
- Real-time coworking capabilities for experiment tracking, feature engineering, and model management
- Controlled model transitioning, deployment, and monitoring

As illustrated in Figure 2-1, MLOps automates the operational and synchronization aspects of the ML life cycle.

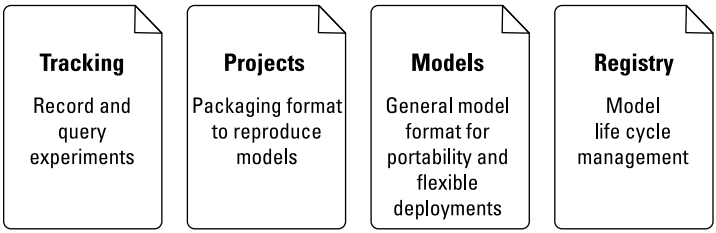


FIGURE 2-1: Automating the ML life cycle.

Borrowing from DevOps

As you learn more about MLOps, you may get a sense of déjà vu. Does it get you thinking of DevOps, the general practices that bring together IT operations and software development?

REMEMBER

You're right. MLOps, in fact, borrows from the more widely adopted DevOps principles in software engineering, even as it applies specifically to ML projects.

DevOps has been around for decades, giving teams a rapid and continuously iterative approach for shipping applications. But DevOps practices and tooling aren't sufficient for the ML world, which is full of models and data and code that require special treatment.

Generally speaking, MLOps borrows principles from DevOps in order to bring ML models to production. But an MLOps solution will take into account the various people and processes that interact with that constellation of artifacts particular to ML applications. It pays attention to specific things such as model accuracy monitoring, data drift, and audit/explainability.

With both DevOps and MLOps, the end result is higher software quality, speedier releases, and faster patching and releases. Consequently, you end up with greater customer satisfaction.

Following the MLOps Guiding Principles

No need to repeat just how complex ML processes are, and how disparate the personas involved tend to be. That means it really isn't possible to put forth an exhaustive set of reference architectures and implementation details, nice as that might be.

REMEMBER

The answer, instead, is to establish broadly applicable principles that can guide MLOps decisions. These principles inform design choices, and the intent is for you to be able to adapt them to support your own particular business case, whatever that might be.

Keeping goals in mind

Take a step back for a moment and remember that the core purpose of ML is to help your business make data-driven decisions and create data-driven products and services. It follows, then, that MLOps is here to ensure those data-driven applications remain stable, that they remain up-to-date, and that they keep on making positive impacts for the business.

Those positive impacts are why you delve into ML in the first place, so you always need to keep them in mind when prioritizing the technical work related to MLOps. Does this work enable a new

business use case? Does it improve the productivity of your data teams? Does it reduce operational costs or risks? If not, is it really a priority?

Taking a data-centric approach

Your aim is to make feature engineering, training, inference, and monitoring pipelines just as robust as any data engineering process should be. What you don't want are tools that make it hard to join data you get from ML predictions and model monitoring and the like with the rest of your data.


TIP

How do you achieve this? Why not develop ML applications in the same place you manage production data? You could, of course, download training data to a laptop, but that makes it hard to govern and reproduce results. You could, instead, secure the data in cloud storage, and make that storage available to the training process.

ML models are essentially data products, and much of MLOps is really about data ops. You need ML tooling that's data-native and able to easily access data both in the lab and production.

Going modular with your implementation

You can't successfully implement any software application without quality code. And indeed, code quality is vital for ML applications, too.


TIP

That's why modularized code is a good plan. It enables you to test individual components, and it mitigates any difficulties you might have with future code refactoring. To clarify the modular structure of your ML application:

- Define clear steps, such as training, evaluation, or deployment.
- Define super steps, such as the training-to-deployment pipeline.
- Define the responsibilities of each person or team for each step.

Letting process guide automation

Should you automate every step of a process? No. Not every process should be, or even can be. Automating processes is for the purpose of improving productivity and lowering the risk of human error, but the fact is, people still determine the business question. For some models, human oversight is essential before deployment.

REMEMBER

For that reason, the development process is primary. Each module in the process should be automated as needed. That allows you to build out automation and customization incrementally.

Also, when considering automation tools, you need ones that align to your people and your processes. For example, should you build a logging framework around a generic database? How about picking a specialized tool such as MLflow, which has been designed specifically for the ML model life cycle? MLflow is an open-source platform for managing ML workflows. It is used by MLOps teams and data scientists. I cover this concept throughout this book, so keep reading to find out more about how MLflow fits into your organization's ML success story.

IN THIS CHAPTER

- Working at an ever-increasing scale
- Cutting out waste
- Bringing teams onto the same page
- Making models sustainable
- Getting better at implementation, iterating, and reproducing
- Maintaining security and solid documentation

Chapter 3
Leveraging the Benefits of MLOps

No one has to tell you how complex machine learning (ML) is and how complicated the ML landscape and life cycle are. This is most certainly not easy work, but the results can be absolutely remarkable. Any organization that tastes the success of ML is going to want more, then more, and still more.

This chapter explains some of the ways machine learning operations (MLOps) can help your organization navigate that complicated environment successfully. It outlines benefits including the ability to scale more efficiently, cut waste along the way, and help your disparate teams collaborate.

As this chapter highlights, MLOps helps you build and train sustainable models, improve how they are implemented, ensure that they are reproducible and secure, and generate audit trails proving that your models are compliant with internal and external requirements and expectations.

Scaling Effectively

Each and every ML model is just as complex as the last one. It never really gets any easier. Your data scientists certainly build experience, but that doesn't mitigate the sheer volume of work and levels of attention that each model requires. You need to be able to scale this work effectively in order to quench the organization's thirst for ML.


REMEMBER

That's one of the most obvious benefits of MLOps. It takes the magic of ML development and implementation, gives it structure, and puts it on a production line. MLOps makes it possible to ultimately develop, oversee, control, manage, and monitor thousands of models simultaneously.

Reducing Waste

We've already noted that the creation of ML models takes time and lots of effort. But is all that time well-spent? Are all the dollars well-invested? Not necessarily.

Whenever you're spending a lot of time on something and exerting a lot of effort, there's a good chance some of that time and effort will be wasted. There's a lot of potential for waste in this particular pursuit, and reducing that waste is a key benefit of MLOps.


REMEMBER

Through the use of standards, tools, processes, and methodology, MLOps will help you to minimize time and money wasted on work that is ultimately abandoned, on efforts that are misguided, on ventures that turn out to be irrelevant for solving a business need. Careful planning and scoping, well-crafted experimentation, development following best practices — it's all key to driving waste out of the ML life cycle.

Encouraging Collaboration

This is, indeed, a team sport, beyond the capabilities of any one kind of data expertise. The pursuit of ML model development and deployment requires midlevel data engineering skills, abilities in software development, and some very capable project management.

Your work will bring together data engineers, data scientists, and machine learning engineers. But you also need business stakeholders, because they're the ones who will be using the models you're developing, not to mention the ones who are establishing the goals and needs in the first place. And data governance officers play a role, too, given the high value of the data you're using and the complex compliance regulations that are likely to be involved.


REMEMBER

Ultimately, there must be many cooks in the kitchen. Keep in mind that when you normally read those words, it's in a negative context — as in "too many cooks in the kitchen." MLOps ensures that your cooks collaborate effectively, not just on that one gourmet masterpiece suitable for the cover of a foodie magazine, but on every meal that kitchen must turn out for a dining room packed with hungry stakeholders.

Building and Training Sustainable Models

A lot of interdependencies are required to make ML happen. A wide range of people, along with a long list of processes and steps. Regulatory and ethical requirements add to the mix.

And these interdependencies change over time. Data that had been a reliable signal eventually becomes little more than noise, new people are on the team, the regulatory climate evolves, open source libraries are no longer fresh enough. You know that things always change — to be sustainable, your models and data and code must change to keep up, and your systems must be resilient.


REMEMBER

MLOps is all about sustainable development, keeping it going at a constant and effective pace. It boosts a number of measures that are vital for sustainability, many of which are listed among the benefits here, from consistency and repeatability to maintainability and availability.

Improving Implementation

Developing and deploying ML models has in the past involved a long series of mostly manual processes. Multiple problems result from overreliance on manual processes. For one thing, manual steps are nearly always slower to complete when compared with more systematic and automated processes. They also are more

prone to error, and that can have a negative impact on models, data, and code. Too much manual work and your ability to get the most out of your data is seriously hampered.


REMEMBER

MLOps reduces the reliance on an overabundance of manual processes. Removing bottlenecks improves the efficiency of implementation, and reducing the potential for human error improves the quality.

Iterating Rapidly

Data scientists and their collaborators must be able to iterate rapidly on ML pipelines and models. For development, quick iteration improves efficiency, and for maintenance, quick iteration allows ML applications to adapt to changing data and user or system behavior. A good MLOps setup allows stakeholders to make adjustments and see the results quickly.

Ensuring Reproducibility

With MLOps, you can package modeling code, dependencies, and all of the various runtime requirements in a way that enhances reproducibility. It's an attractive alternative to the error-prone manual processes that have traditionally been part of packaging and deploying ML solutions.


REMEMBER

You're reducing the potential for disconnects and misunderstandings between the data scientists that build the models and the software engineers that deploy them. And you're reducing the work required when the modeling framework has to be updated.

A reproducible ML approach makes it simpler and less costly to package and maintain model versions, and it helps your organization to more effectively deploy at scale.

Maintaining Security and Compatibility

Any kind of work involving vital data must be protected with absolutely rock-solid security. The ML life cycle is certainly no exception. MLOps ensures that your teams aren't inadvertently

missing steps that could compromise security. And the choice of the right tools further solidifies that protection.


REMEMBER

For example, Databricks provides a fully managed and hosted version of MLflow. The open-source MLflow manages the ML life cycle, including experimentation, reproducibility, deployment, and a central model registry. When you opt for the Managed MLflow on Databricks, you gain extra enterprise security and management features.

Creating Audit Trails

You can have a great ML solution up and running, and still wind up with a failure in the win-loss column if you're not careful. Your solution must not only work, but also must have auditable proof that it's working as planned, and in compliance with internal and external requirements and regulations.

Indeed, a well-designed system that can't demonstrate compliance with corporate, regulatory, or ethical requirements could end up generating fines, causing reputational damage to the organization, or losing its funding. And if any of those bad things happen, can you really say it was a well-designed system?


REMEMBER

MLOps helps to ensure that you're not lacking when it comes to documentation, model monitoring, and KPIs. You should have full information on everything from parameters, metrics, and model artifacts, through all of a model's life cycle stage transitions.

Enabling Training and Retraining

Training and retraining are key components to the MLOps approach. Depending on the model deployment pattern you choose, model training may take place in the development environment, or in the development and staging environments, or potentially in the production environment, too.


REMEMBER

Which approach you choose depends on a number of factors, including the costs of training in each particular case, and whether security considerations will allow production data to be accessed in earlier environments. MLOps helps you to chart the right path and follow it to success.

IN THIS CHAPTER

- Analyzing and preparing data
- Training your model
- Reviewing and establishing governance
- Deploying and monitoring models
- Planning for model inference and serving

Chapter 4
Discovering MLOps Components

What are the building blocks of machine learning operations (MLOps)? It depends a bit on the individual project, its size, its scope, and its demands. But there are some common components into which the MLOps approach can be classified. You can employ MLOps in a narrowly focused way in your ML projects, or it can be expansive from the start or grow with the project's demands. MLOps may cover everything from the data pipeline through model production. In other cases, you may need MLOps implementation for just the model deployment.

This chapter explores the primary components of deploying MLOps principles. Take a look at Figure 4-1, which shows you a typical ML workflow detailed with the steps in which enterprises can employ MLOps. You discover each step in Figure 4-1 in this chapter.

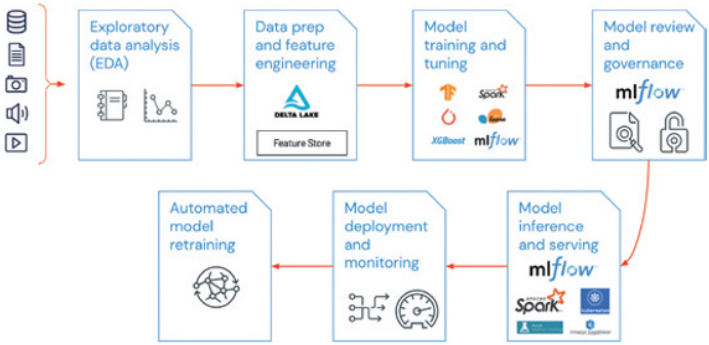


FIGURE 4-1: Deploying MLOps principles.

Exploratory Data Analysis

Within this component of the ML process, data scientists analyze the statistical properties of the data available. Their aim is to figure out if the data will address the business question. They'll need to be in close contact and iteration with business stakeholders.

REMEMBER

In the context of MLOps, I'm talking about iteratively exploring, sharing, and prepping data for the ML life cycle. That requires creating reproducible, editable, and shareable datasets, tables, and visualizations.

Data Prep and Feature Engineering

Before one can delve into data science or ML work, data engineers must prepare production data and make it available for consumption. Without the data, nothing else of importance can happen. Feature engineering refers to the work data scientists do to clean input data and apply any business logic or specialized transformations needed for model training. They create data sets for training, testing, and validation.

TIP

I'm talking about iteratively transforming, aggregating, and de-duplicating data to create refined features. It's vital to make the features visible and shareable across data teams, making use of a feature store.

Model Training and Tuning

When it comes to model training, data scientists explore multiple algorithms and hyperparameter configurations using the prepared data. Their goal is to determine the best-performing model that meets a set of predefined evaluation metrics.

Before it's deployed, the selected model runs through a validation step to be sure it performs above the established baseline level of performance. And it needs to meet all other requirements — whether technical, business, or regulatory. Figuring all that out requires close collaboration involving data scientists, business stakeholders, and ML engineers.

TIP

In the world of MLOps, this component taps into popular open-source libraries for training and improving model performance. You can also use automated ML tools such as AutoML to perform trial runs automatically and come up with reviewable and deployable code.

Model Review, Explainability, and Governance

The ability to review and govern models and to explain their behavior benefits all ML projects, and it's required by regulation in some industries. Tracking model lineage and versions lets you know exactly which model is in production and how it was produced. Explainability tools let you examine a model to understand why it makes certain predictions.

REMEMBER

An open-source MLOps platform, such as MLflow from Databricks, simplifies these key elements of ML. MLflow enables your team to discover, share, and collaborate on ML models.

Model Inference and Serving

This component of MLOps is all about making predictions with your model for end-users and applications. Your inference and serving system may provide predictions to downstream systems via published tables of predictions, streams of predictions, or REST APIs.

REMEMBER

Some applications will benefit from batch or streaming inference, which is high-throughput and low-cost. Other applications will require low (sub-second) latency predictions, which can be provided via online serving.

Model Deployment, Monitoring and Observability

Your ML engineers deploy the validated model by way of batch, streaming, or online serving. Which approach depends on the particular use case and its requirements. Once models are deployed, ML engineers continue to monitor them, watching for any indications of errors or signs that performance is degrading. If such issues are spotted, that may mean it's time to update the deployed model.

The MLOps process can automate testing of new models against current production models, deploying new models to serving clusters and REST API model endpoints, and monitoring the predictions after deployment. Ultimately, you are able to keep tabs on the availability of the model, its predictions and performance on live data, and the performance of the ML system from a computational perspective.

REMEMBER

The MLOps approach takes advantage of continuous integration and continuous deployment (CI/CD) tools, such as repos and orchestrators to automate the pre-production pipeline. This taps into DevOps principles.

Automated Model Retraining

Are you experiencing model drift because of differences in training and inference data? It's inevitable because data keeps on changing, and simply deploying a model in production doesn't mean it's going to work indefinitely.

REMEMBER

With MLOps, you can create alerts and automation that takes corrective action. Like pretty much all of us humans, an ML model is going to need retraining at some point. But unlike with humans, retraining in the ML world can be automated.

IN THIS CHAPTER

- Meeting the ML team
- Working in the various execution environments
- Pulling together the assets
- Lining up the MLOps workflow

Chapter 5
Managing Your MLOps Team and Workflow

A machine learning (ML) project brings a lot of cooks into the kitchen. That's a reasonable analogy, but it may be that an orchestra is an even better one. Many different players are involved in this performance, and their work can either sound like a symphonic masterpiece or a cacophony.

Machine learning operations (MLOps) is, in a way, like a combination of the musical score and the conductor. It spells out the big picture with all of the roles and processes, then helps to orchestrate it all nicely. This chapter offers some description of the players and the parts for which each is responsible, as well as the workflows that lead the symphony from the opening note through the repeat sign guiding the musical journey back through the beautiful process again.

Gathering the Players

It's barely short of miraculous the work that your ML models can do on their own once they're deployed. But independent as they may seem when they're doing their job, it still takes plenty of human hands to get them to that point and keep them running successfully.

Indeed, the work of building ML applications is a team sport. It's worth running through the cast of characters to gain a full understanding of who is responsible for what.


REMEMBER

Of course, in any real-world environment, you have people who wear multiple hats. Some of the different functions may be managed by the same person. But for now, I spell them out in terms of archetypes, to appreciate the different roles and responsibilities, where tasks typically hand off from one role to another, and just how complex this work can be.

Data engineers

Data engineers have responsibility for building data pipelines. These pipelines allow ML apps and other downstream applications to process, organize, and persist datasets.

Data scientists

These players on the team have the job of gaining an understanding of the business problem and exploring available data to determine whether ML is a potential solution. If the answer to that question is "yes," your data scientists will train, tune, and evaluate the ML model before it's deployed.

ML engineers

Your ML engineers deploy these ML models into production. Among the responsibilities of these team members are ensuring that you have in place appropriate governance, adequate monitoring, and exceptional software-development best practices. Those best practices include continuous integration/continuous deployment (CI/CD).

Business stakeholders

The business stakeholders may not be doing the behind-the-scenes work on ML models. But they're the ones who use the model to make decisions for the business or product. Creating business value is their responsibility, and the ML model is among the ways they will generate that business value. Their role is to specify the work and goals you're aiming to achieve through an ML model, and then ensure that the solution is delivering on its promises.

Data governance officer

This role is not necessarily involved in day-to-day operations, but it's an important part of the ML success story. The data governance officer ensures that data governance, data privacy, and other compliance measures are maintained and followed across the development and deployment of the ML model.

Operating Within Your Execution Environment

Simply put, the execution environment is the place where your models and your data are created, or where they are consumed by code. Environments vary significantly with regard to what's happening within them, what the quality guarantees are, the level of risk, and the access control.


REMEMBER

The *dev* environment — dev is short for development — is where many data scientists spend a lot of their time working together, freely producing their dev model prototypes. In our orchestra analogy, this could be the room where the composer is working, charting out the symphony.

These model prototypes are separate from the live product, which means any flaws within them are relatively low-risk. Data assets in this environment are widely accessible to more people, but there's less guarantee of quality.


REMEMBER

The most promising models from the dev environment graduate to the *staging* environment. This environment is a mirror of production, and models are tested here prior to release. For the orchestra perspective, this is somewhat akin to the rehearsal room, or perhaps the stage in front of an empty auditorium.

Fewer people have access to the staging environment for a number of reasons. For one thing, incorrect changes made here can make the model deviate from production behavior, which would raise the risk that bad models could be promoted to production.


REMEMBER

Models that succeed in the staging environment are ready to promote to the *production* environment — and that is where they become live product. This is the concert hall, with the seats filled with audience members anticipating a great performance.

Human error in this execution environment could be especially catastrophic and a major risk to business continuity. For that reason, only a limited number of people have access to modify production models.

The various execution environments consist of compute instances, their runtimes and libraries, and automated jobs. It's worth noting that an environment may be defined in terms of dev and staging and prod at a few levels. Your organization could have distinct environments across multiple cloud accounts, or multiple workspaces within the same cloud account — or within a single Databricks workspace. Take a look at Figure 5-1 to see how these separation patterns may work.

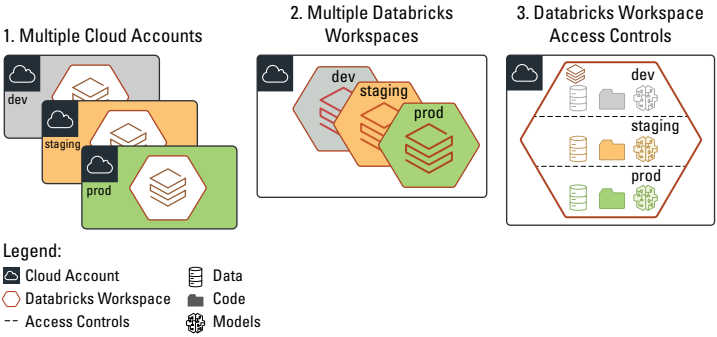


FIGURE 5-1: Three environment separation patterns.

Assembling Key Assets

Three key kinds of assets make up ML workflows: code, models, and data. All of these assets will need to be developed (or *dev* for short), tested, and deployed into production (typically shortened to *prod*).

Code

You often store your ML project code in a version control system such as Git. Most enterprises will use branches that correspond to the life cycle phases of development, staging, and production.

But, of course, nothing in life is quite that simple. For example, some will split the code into only two branches, one for dev and

one for staging and production. Others may opt for main and development dev branches, plus staging branches cut for testing potential releases, and then prod branches cut for final releases. In any case, this separation is enforced through Git repository branches.


TIP

Here's a best practice to keep in mind: You should only run code in an execution environment that corresponds to it or higher. You could, for example, run any code in the dev environment, but you should only run prod code in the prod environment.

Models

Models also tend to be marked dev, staging, or prod, depending on their life cycle phase. But here's another place where things get a little murky. Model and code life cycle phases often operate asynchronously. What that means is, you may want to push a new model version before you push a code change, or the other way around.

To help explain that, consider these examples:

- Say your aim is to detect fraudulent transactions, and you develop an ML pipeline that retrains a model weekly. You may not deploy code all that frequently, but every week there's a new model with its own life cycle of being generated, tested, and marked "production" to predict on the most recent transactions. You have a situation where the code life cycle is slower than the model life cycle.
- Imagine you're classifying documents using large deep neural networks. Training and deploying the model is often a one-time deal, because it's so costly. There may be updates to the serving and monitoring code that are deployed more frequently than a new version of the model. So, you've got a model life cycle slower than the code life cycle in this case.


TIP

Given this asynchronous situation between the model and code life cycles, you're going to want model management to have its own service. MLflow and its Model Registry, for example, support managing model artifacts directly by way of UI and APIs.

Through this kind of loose coupling of model artifacts and code, you gain a lot of flexibility and can streamline the deployment

process in a lot of cases, updating production models without code changes. You secure these model artifacts by using MLflow access controls or cloud storage permissions.

Data

So how do you categorize your data? One good approach is to label it as either dev, staging, or prod, depending on the environment where it originated. Therefore, all prod data, for example, is produced in the prod environment, and your dev and staging environments have only read-only access.


REMEMBER

If you use this approach, you'll ensure that the guarantee of data quality is explicit. Dev data, for example, may be temporary or not meant for wider use, while prod data is likely to have stronger guarantees with regard to freshness and reliability. You should control access to data in each environment with table access controls or cloud storage permissions.

The point is that in MLOps you always maintain operational separation between dev, staging, and prod. On the dev side, your assets will have the least restrictive access control and lower quality guarantees. Your highest-quality assets will be found in production, and they'll be the most tightly controlled.

Preparing Your MLOps Pipeline

There are multiple ways to get your ML artifacts through the process, from development to staging and on into production. The two most prominent deployment patterns differ in terms of which element is promoted toward production: the model artifact, or the training code that produces the model artifact. See Figure 5-2 for a visual depiction of the two possibilities.

That first pattern depicted in the diagram is the deploy models option, with a model artifact generated in the dev environment. That artifact then gets tested in staging for both compliance and performance, and it gets deployed into production once it passes the tests.


REMEMBER

It's a simpler way to roll, for sure. And it may be preferable in cases where model training is super-expensive. This way, you train the model once and then manage that artifact.

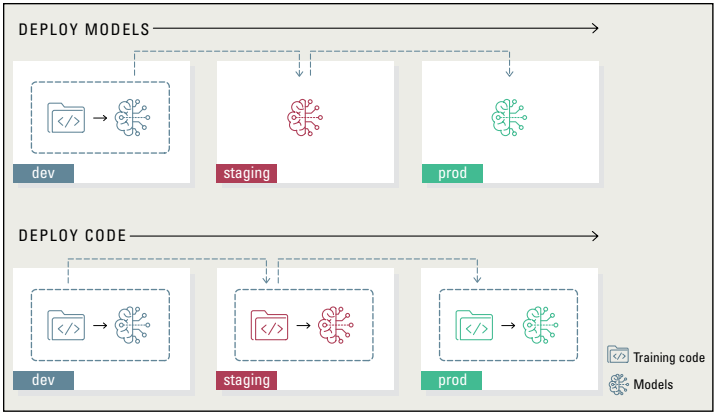


FIGURE 5-2: Major deployment patterns.

Simple isn't always better, though, and this approach has some drawbacks. For example, if for security reasons you can't work with production data in the dev environment, you may wind up with an architecture that's not viable. Also, this architecture isn't ideal for automated model retraining. You can do that in the dev environment, but you'd be acting as though dev training code is production-ready, and your deployment teams may not be keen on that. There's also ancillary code for featurization, inference, and monitoring, all of which has to be deployed to production. That'll require a separate deployment path.

REMEMBER

Moving on to the second pattern deployed in the figure, referred to as the deploy code pattern, you'll use the dev environment for creating the code for training models, then move this code to staging, then production. That's represented by the image of folders containing training code, showing up in all three environments.

What that means is you're training models in each environment. It happens first in the dev environment as you're developing the model, then in staging as part of integration tests on limited subsets of data, and then using the full production data to produce the final model in the production environment.

In situations where data scientists have restricted access to production data from the dev or staging environments, this pattern lets you train on production data while still adhering to those access controls.

Because training code is run through code review and testing, it's safer to establish automated retraining. Your ancillary code will be following the same pattern as the model training code, and there can be integration testing in staging for both.


TIP

A few things to keep in mind about this pattern: It's wise to create opinionated project templates and workflows, because many data scientists will find there to be a steep learning curve when it comes to handing code off to collaborators. Also, be sure those data scientists have visibility into the production environment training results — they're really the only ones with the knowhow for spotting and fixing ML issues there.

So which approach is better? It depends on the business use case and the resources you have available. There are some situations that might call for deploying model artifacts and other cases where deploying training code makes the most sense. That said, the deploy code approach is generally the most suitable the majority of the time (check out Chapter 6).

IN THIS CHAPTER

- Introducing the workflow components
- Understanding the reference architecture
- Going through development, staging, and production

Chapter 6
Running through the Reference Architecture

Now comes time to see how machine learning operations (MLOps) works in practice, using the features provided on Lakehouse platforms such as Databricks and tapping into the open-source MLflow management environment.

This chapter introduces the components of the reference architecture and then takes a run through all the MLOps steps in greater detail. You have choices when it comes to your pattern for deployment — deploying models or deploying code (see Chapter 5). For the sake of simplicity, this chapter focuses on the deploy code pattern, which is best for the majority of use cases.

Getting to Know the Components

REMEMBER

A great place to dive into making MLOps work for your enterprise is becoming familiar with the components that play a role in the workflow. Specifically, here are Databricks features and related elements that facilitate MLOps.

- **Data Lakehouse:** This architecture unites the best parts of a data lake and a data warehouse. It brings together low-cost, flexible object stores like you get in a data lake, along with the kind of data management and performance you expect from a data warehouse.
- **MLflow:** This open-source project manages the machine-learning life cycle from end to end, and a fully managed and hosted version of MLflow is integrated into Databricks offerings. MLflow lets you track experiments to record and compare parameters, metrics, and model artifacts. It lets you store and deploy models from any ML library to model serving and inference platforms. And it includes a Model Registry, a centralized model store for managing your models' life cycle stage transitions.
- **Autologging:** Databricks Autologging extends MLflow's automatic logging for experiment tracking.
- **Feature Store:** This centralized repository of features enables feature sharing and discovery, and helps you ensure you're using the same feature computation code in training and inference.
- **MLflow Model Serving:** Host ML models from the Model Registry as automatically updated REST endpoints.
- **Databricks SQL:** This makes it easier for SQL users to run quick ad hoc queries on their data lake, as well as create dashboards and alerts for monitoring.
- **Databricks workflows and jobs:** These are tools for executing pipelines in automated and non-interactive ways, and define pipelines for computing features, training models, and the like.

Exploring The Reference Architecture

REMEMBER

To continue down the path of understanding the MLOps approach, in this section, you take a stroll through the environments within the reference architecture in MLOps. I spell out the overall process for deploying code and model artifacts, starting in development, graduating up to staging, and then promoting into production.

Working in the dev environment

REMEMBER

The development (dev) environment is where data scientists and ML engineers collaborate on all the pipelines of an ML project, committing changes to source control. In Figure 6-1, you see how tasks flow within dev.

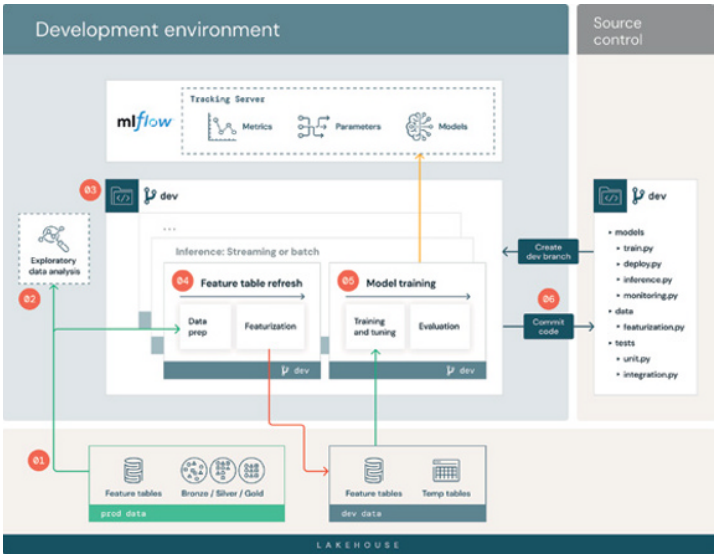


FIGURE 6-1: The dev environment.

For details on the different tasks within the dev environment, check out this list:

- **Data:** In the dev environment, data scientists have read-only access to production data, or a filtered or obfuscated dataset if needed for regulatory reasons. They need a separate dev storage environment with read-write access where they can experiment with and develop new features and data tables.
- **Exploratory data analysis (EDA):** This interactive, iterative process is for exploring and analyzing data. Will the available data address the business problem? Here's where the data scientist gets a sense of that, and if so, begins to figure out what kinds of data prep and featurization will be needed for model training. The processes explored here won't be deployed in other execution environments.

- **Project code:** A code repository contains all modules or pipelines in the ML system, with dev branches used for developing changes to pipelines or creating new ones. It's best to develop within a repository so you can share code and track changes.
- **Feature table refresh:** This pipeline is for reading from raw data and feature tables and writing to the tables in the Feature Store. Data prep finds and addresses data quality issues first, then featurization tests new features and updated featurization logic. The process is to write to feature tables in dev storage, then use those tables for model prototyping. When promoting featurization code to production, the changes affect production feature tables.
- **Model training:** In the deploy code approach, you develop the model training pipeline in the dev environment, using dev or prod feature tables:
  - **Training and tuning:** In training, you're reading features from the feature store or Lakehouse tables, and logging model parameters and metrics and artifacts to the MLflow tracking server. The final artifact is logged to the tracking server to record a link between the model, input data, and code.
  - **Evaluation:** Here's where you evaluate model quality, testing it on held-out data and logging results to the MLflow tracking server. You can also track any additional metrics or documentation required by governance.
  - **Model output:** What you get from this process is an ML model artifact stored in the tracking server. ML engineers or continuous integration/continuous deployment (CI/CD) code can load the model and push it to the Model Registry for management and testing.
- **Commit code:** The ML engineer commits dev branch changes into source control.

Moving to staging


REMEMBER

Your transition of code from dev into production makes a stop in the staging environment. ML engineers manage continuous integration pipelines and orchestration, and they and data engineers write tests for code and models. Figure 6-2 gives you a visual idea of how this goes.

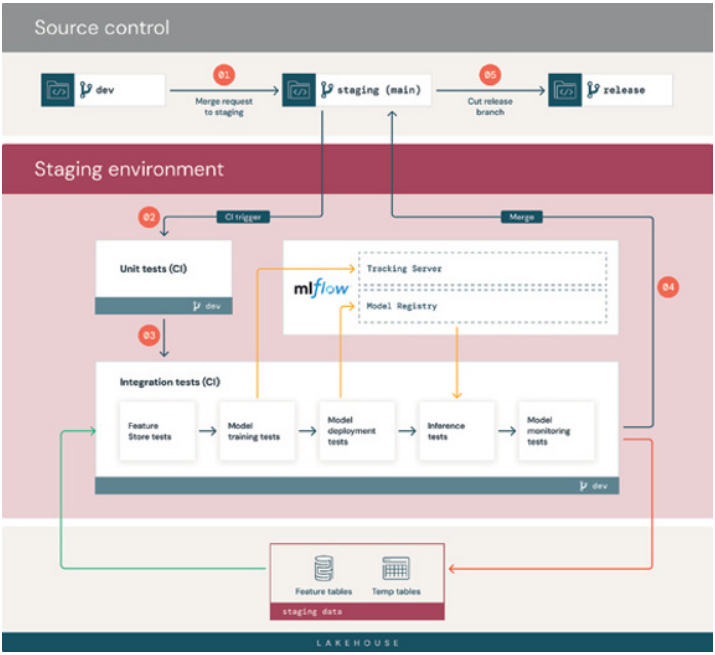


FIGURE 6-2: The staging environment.

Here are more details about staging:

- **Data:** In staging, there may be a storage area for testing feature tables and ML pipelines. Data is usually temporary, retained only long enough for running tests and investigating any issues. You can also make data readable from dev for debugging.
- **Merge code:** A merge or pull request gets the deployment process going, submitted against the staging branch of the project in source control. The merge request builds source code and launches unit tests. If those tests fail, the merge request is rejected.
- **Integration tests:** Also referred to as continuous integration or CI, this runs pipelines to confirm they work well together. This environment should mimic production as much as possible. If models are expensive to train, you may test model training on small data sets or fewer iterations in order to save money. If tests pass, new code is merged into the staging branch.
- **Cut release branch:** After these CI tests pass on a commit in the staging branch, ML engineers cut a release branch.

Heading into production

REMEMBER

A select set of ML engineers manage the prod environment, which is where ML pipelines directly serve the business or application. In these pipelines, fresh feature values are computed, new model versions are trained and tested, predictions are published, and the process is monitored. Check out Figure 6-3 for a look into this environment.

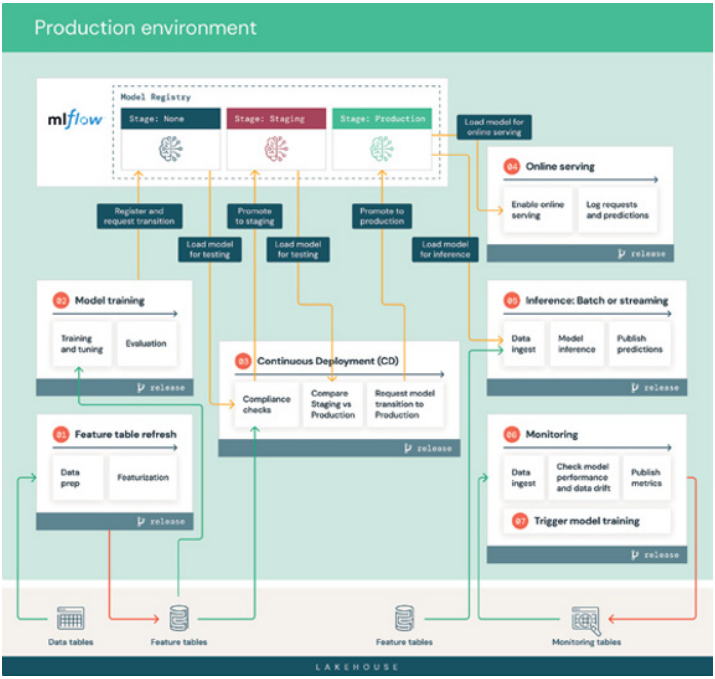


FIGURE 6-3: The production environment.

The production tasks are as follows:

- **Feature table refresh:** The pipeline transforms Lakehouse data into production feature tables. It can use batch or streaming computation, depending on the freshness requirements for training and inference. Within Databricks, the pipeline can be defined as a Databricks Job.

- **Model training:** When code changes impact upstream featurization or training logic, or if automated training is triggered, the model training pipeline runs.
  - **Training and tuning:** Logs are recorded to the MLflow tracking server, including model metrics, parameters, tags, and the model itself. Only the top-performing algorithms and hyperparameters should have made it into production training code.
  - **Evaluation:** Here you evaluate quality by testing on held-out production data and logging the results in the tracking server. Data scientists should establish metrics for this during development.
  - **Register and request transition:** After training, the model artifact is registered to the MLflow Model Registry in the production environment. The final step is to request a transition of the newly registered model to "stage=Staging."
- **Continuous deployment:** At this point, three CD tasks happen:
  - **Compliance checks:** The model is loaded from the Model Registry to check tags, documentation, and the like. This is an automated step, but it can create statistics or visualizations if human review is needed. Results are logged in the Model Registry.
  - **Compare staging to production:** It's important to compare those models promoted to "stage=Staging" with the ones they are destined to replace that are marked "stage=Production." That's how you prevent performance degradation. On the first deployment, there's no existing prod model to compare with. In that case, you can compare the "stage=Staging" model with some other baseline or business heuristic.
  - **Transitioning to production:** Now it's time to transition to "stage=Production" in the Model Registry. Like all stage transition requests, this can be done manually through the MLflow user interface, or automatically through APIs and webhooks.
- **Online serving:** This is usually necessary for use cases with lower throughput and latency. You can deploy models to Databricks Model Serving, or cloud provider serving endpoints, or on-prem or custom serving layers.

- **Inference:** This pipeline is for reading data from the Feature Store, loading the model, performing inferences, then publishing them. For a batch job, you'll likely publish predictions to Lakehouse tables, flat files, or over a JDBC connection. If streaming, predictions could go to Lakehouse tables or message queues.
- **Monitoring:** In MLOps, you monitor input data and model prediction for statistical properties and computational performance. This pipeline ingests data, checks for accuracy and data drift, publishes metrics, and if it is determined to be necessary for performance or freshness reasons, it triggers model retraining.
- **Retraining:** This is the final step in the prod process, and in the architecture described, it can happen automatically using the same model training pipeline as used before. It's best to start with manually triggered retraining, though, then work your way up to scheduled or triggered retraining. Scheduled retraining happens when you anticipate and need fresh data on a regular, periodic basis. Triggered retraining happens when the monitoring pipeline finds there are performance issues or significant changes in the distribution of incoming data.

IN THIS CHAPTER

- Simplifying the ML process
- Creating real business value
- Personalizing the customer experience
- Making better business decisions
- Reducing fraud and enhancing security

Chapter 7
Ten Real-World Use Cases

How does machine learning operations (MLOps) bring about enhanced business value, improved operations, and greater customer experiences? This chapter gives you ten (okay, there's only seven, you caught me) examples of real-world use cases that were empowered by Databricks and MLOps.

Enabling Simplicity

MLOps, with the help of Databricks products, has greatly simplified the work of an Australian company that creates software development and collaboration tools. The company was thrilled to compress the number of steps between writing and testing local code, to pushing it to the cloud.

This company's data scientists found that the overall iteration cycle time reduced from weeks to days. Organized machine learning (ML) tracking with MLflow brought together all lineage and logging for parameters, code versions, metrics, artifacts, and other ML activities. That makes it possible to look over results and choose the best possible option, and it facilitates reusability and shared learnings. Multiple teams examine prior versions and

their metrics, then decide to reuse existing models or make new ones. And because MLOps makes it so much easier to implement ML models, teams beyond data science are using it for their own specific use cases. That's self-service!

Focusing on the Business Value

It's all too easy to get lost in code and numbers, according to data science experts from an American supply chain management, transportation and logistics company. That's why it's vital for data scientists to never lose sight of the need to create business value and accomplish objectives of the organization. The company cites a number of guiding principles that have enabled this focus and have made its MLOps journey a success:

- **Automated:** Users don't have to remember all the rules for acting and implementing.
- **Secure but self-service:** Users are empowered but still within boundaries designed to keep them safe.
- **Repeatability:** Configuration artifacts follow the code throughout the ML life cycle.
- **Clearly defined environments:** It's not just automating existing workflows; it's also giving tools to create new possibilities.
- **Platform and cloud agnostic:** It's important to be able to produce solutions that can run anywhere.

Reaping the Benefits of ML

A global independent consultancy that specializes in data science, data engineering, and data visualization reports that MLOps really opens the door to the vast benefits of ML at scale. Here are some of those benefits:

- **Scalability:** ML can be scaled efficiently both horizontally and vertically, with out-of-the-box optimizations lowering infrastructure costs.

- **Reproducibility:** MLOps saves time and creates governance for product teams, using tools that enable reproducibility of experiments and model training.
- **Reusable assets:** Tracking, monitoring, and identifying reusable assets helps increase efficiency and cost savings.
- **Automated model retraining:** This decreases manual dependencies, with pipelines configured to trigger automated retraining.
- **Model evaluation:** This company is able to maintain and monitor model quality with the help of standardized and consolidated KPIs and evaluation metrics.
- **Fast feedback loop:** MLOps helps the company respond quickly to business opportunities and changes.

Making ML Fashionable

A major clothing retail chain has an artificial intelligence (AI) team of more than a hundred people working on building systems for production. The ML-driven tools this team has developed tackle all kinds of incredibly valuable tasks, including quantifying the inventory that should be made available to customers, negotiating costs with various supplies, allocating inventory between warehouses, determining selling prices and markdowns, and delivering customers very personalized item recommendations.

The company set out to "democratize" AI across its business technology. MLOps promotes reusability, integrates ML into the value chain, cuts time-to-market in half for AI use cases, and improves processes.

Making Entertainment Personal

A major communications, media and entertainment company has millions of customers but still wants to deliver personalized experiences. ML is a great recipe for that treat, but the company was hampered by a vast amount of data, fragile data pipelines, and poor data science collaboration.

Databricks solutions simplified the life cycle management of hundreds of models, and the result was an award-winning user experience powered by voice recognition. The company cites the following MLOps benefits:

- Simplified infrastructure management
- Performant data pipelines
- Ability to reliably manage small files
- Collaborative workspaces
- Simplified ML life cycle management with MLflow
- Reliable extract, transform, load (ETL) at scale
- Efficient analytics pipelines at scale

Banking on ML

A major bank in the Netherlands wanted to up its game as a data-driven organization, shifting toward the Azure cloud and operationalizing through the Databricks Lakehouse Platform. Its data scientists and ML engineers report a boatload of benefits as they easily deploy models into production with MLflow.

Once-disjointed, manual, and inefficient processes are now consistent and automated. Among the benefits this bank reports are improved strategic decision-making, better operational efficiency, more robust cybersecurity, and an enhanced customer experience.

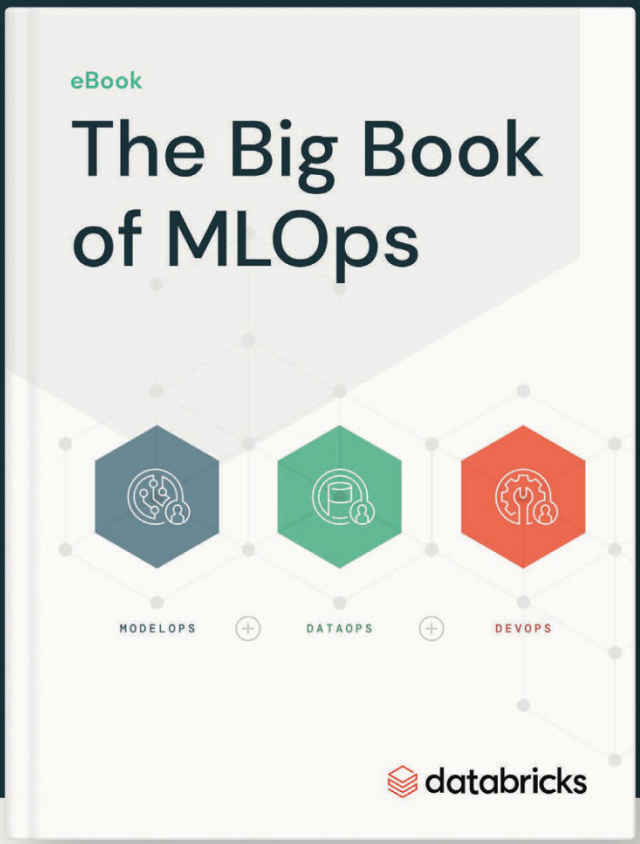
The bank can react to customer preferences faster and serve relevant product recommendations. It has vastly improved fraud detection through the use of ML that spots anomalous behavior that could indicate such things as money laundering. And its systems now power a client dashboard that provides a complete view of customers and all their assets and transactions in near real-time.

Upgrading the Data Journey

Digital transformation was also the goal of a financial services company in Hong Kong. Its journey was driven by data and analytics, upgrading legacy on-premises data warehouse systems that had become structurally difficult to maintain. The company knew better data insights would mean better customer service, and it turned to Databricks for answers.

A centralized Lakehouse supports development of ML models that personalize the customer financial experience while ensuring the privacy and security of customer data. The organization increased operational efficiency, enhanced risk assessment, gained more comprehensive management of information, and experienced double-digit growth in lead generation.

databricks



Get the eBook →

There's a new data-centric approach to scaling ML

Learn how to build — and scale — your machine learning operations. The Big Book of MLOps outlines the people, processes and tools you'll need to succeed.

In this eBook, you'll learn:

- The 7 key steps of ML application development
- 4 guiding principles for your MLOps decisions
- The 5 types of team members you'll need to build ML applications

Maximize machine learning power and speed

Adoption of machine learning has exploded in recent years, driving better analysis and supercharging operational processes. Your enterprise must get good at the complex science — and become fantastic at operationalizing the magic of ML. MLOps is an exceptional approach for upping your game. *MLOps For Dummies,* Databricks Special Edition, helps you scale your operations to manage a *lot* of ML solutions. Open this book to learn effective ways to build collaboration, improve implementation, and reduce risk.

Inside…

- How ML is changing the business world
- Where ML management gets tricky
- Who on your team makes ML happen
- The components that are part of MLOps
- Benefits of adopting an MLOps approach
- MLOps transforms dev, staging, and prod
- Real-world use cases for MLOps

databricks

Steve Kaelble is the author of many books in the *For Dummies* series, and his writing has also been published in magazines, newspapers, and corporate annual reports. When not immersed in the *For Dummies* world or writing articles, he engages in healthcare communications.

ISBN: 978-1-119-98167-1
Not For Resale

for dummies®
A Wiley Brand

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.