Whitepaper

MLOps Beyond the Hype

Concrete Answers to Questions About MLOps

Table of Contents

Why Do I Need MLOps?

The perceived impact of AI is at its peak with the rise of ChatGPT and Generative AI. However, building machine learning (ML) solutions is still costly, and many of these never reach an end user.

There are three main issues that make ML projects fail:

Production gap
While it is easy to create a prototype for a solution, it is hard to build a first version that eventually reaches end users.

Long iteration cycles
Once the first version of a solution reaches end users, long iteration cycles can make it difficult to add new features and ensure stability.

High costs
The combination of the previous factors and a lack of shared standards might result in custom and complex solutions. These are expensive to build and maintain.

MLOps aims to fix these problems by ensuring that teams have the required expertise and tooling to autonomously build and maintain ML solutions. This reduces the production gap and the length of iteration cycles. Further standardization drives down costs even more.

In this whitepaper, we will introduce the practices of MLOps, so your teams may create and iterate on their solutions much faster, saving you time, money, and the frustration of unfinished projects. We will guide you through the necessary changes regarding people, processes, and technology to make MLOps a reality in your company.

“MLOps = DevOps + managing models and bringing them into production in a scalable and sustainable way.”

— Travis Dent, Data/ML Engineer at Xebia Data

Definitions used in this whitepaper

- **DevOps** is a working methodology that combines software development with operations. It creates an environment of collaboration and continuous feedback to ensure the fast development of applications and the minimization of waste.
- **Machine Learning (ML)** is the process that allows a computer to learn rules from example data rather than from explicit instructions. Through continuous training and optimization, the computer creates a model that fits historical data and adds predictions or extracts insights from new data.
- **MLOps** is the combination of ML and DevOps. This discipline reaps the benefits from DevOps and applies them to specific Machine Learning solutions to ensure its rapid and efficient development.
- **Application Programming Interface (API)** is a program or piece of code that allows two computers to exchange information. This process happens through a series of calls from one computer to another to request and edit the necessary data.

Use Case: Generating Product Descriptions for a Classified Ads Platform

To illustrate what MLOps means in practice, we will now introduce a use case that will be used in the rest of the whitepaper.

Let us consider a classified ads platform that allows consumers to sell their second-hand products. A new ML solution should generate product descriptions when users upload pictures of their items (Figure 1). These enhanced product descriptions should result in easier sales for the users and increased revenue for the platform.

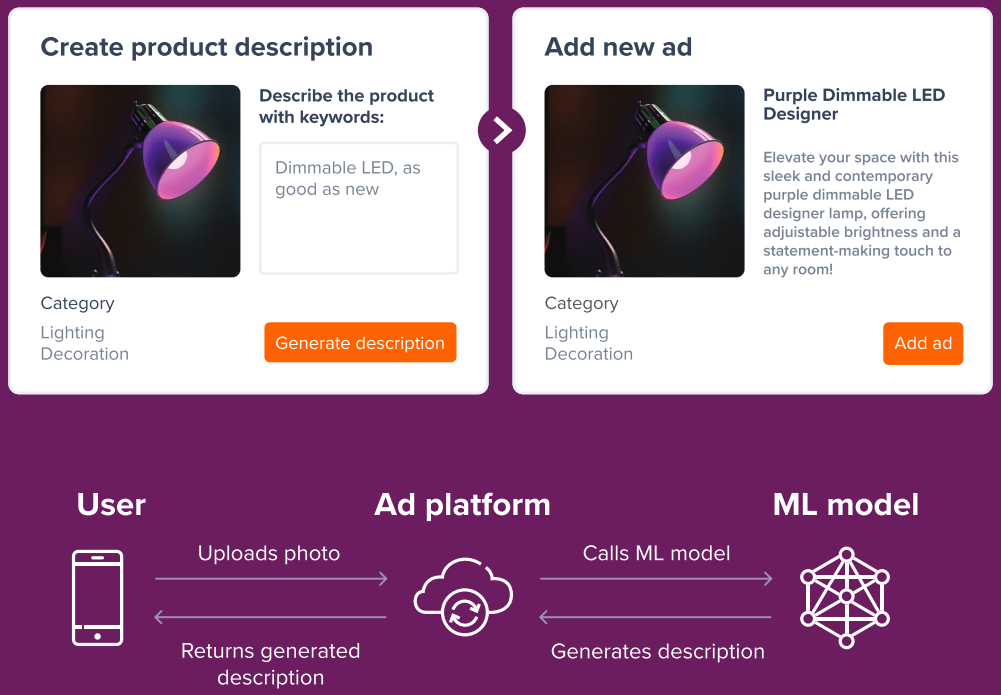

Figure 1: *Illustration of the classified ads use case. Users upload pictures of their items, and an ML solution generates the product description. The Ad platform acts as an intermediary between the user and the model.*

A data scientist develops a first prototype with a generative model that writes descriptions from uploaded pictures. The model is accurate enough but has been developed in a Jupyter Notebook: a virtual scratchpad that doesn't integrate with other services. Before the model can be used in the app, it needs to be turned into a proper solution that can generate descriptions on the fly.

Turning this notebook into a full-fledged solution is not easy. Measuring real-life performance poses the following questions: how can we assess whether the model generates a good or a bad description? How can we detect that our algorithm fails to describe a certain type of shoes? And if we can't measure the accuracy of our predictions, what should we focus on? Even keeping the solution up to date is hard: new products are being added and retraining the generative model is a manual operation.

Building and maintaining this solution is currently costly and time-consuming. What's worse, other teams may have similar problems without anyone being aware of the shared difficulty. Each team is reinventing the wheel and building bespoke solutions because there is a lack of common platforms or ways of working. In this scenario, costs often outweigh benefits.

"Data Science is taking data and building a model for it. But to create value, you need to run the model, monitor its performance, and make sure it keeps making the correct predictions. All of that 'taking care of the model' is MLOps."

— Rogier van der Geer, Data Charmer at Xebia Data

What Does MLOps Solve?

MLOps streamlines the process of building and maintaining ML solutions. It was born as a response to the pains and high costs that companies encounter when shifting their focus from experimenting with ML to developing ML solutions.

Like DevOps, MLOps is characterized by a few key principles:

Autonomy: teams are responsible for a solution throughout its entire lifecycle, from inception to final product.

Rapid feedback and iterative development: teams actively seek feedback and use it to iteratively improve their solution.

Automation and collaboration: teams streamline their workflow with the right tools and best practices.

“MLOps is a logical next step in the ML domain. It’s a maturity thing. Before, users would deploy an ML model in an ad hoc way. Now, it’s becoming more structured, more accessible, and more scalable.”

— Daniel van der Ende, Data Engineer at Xebia Data

To illustrate what this means in practice, let’s consider our image classifier use case for the classified ads platform from the previous section.

In our use case:

Autonomy gives the team control over the solution throughout its entire lifecycle, closing the production gap. The team has the responsibility to develop an ML solution that generates descriptions for pictures uploaded by users, not just a one-off prototype.

Rapid feedback and iterative development enable the team to improve and extend the solution step by step, reducing the length of iteration cycles. Feedback can come in many forms. Content-wise, end users may rate generated descriptions. Performance-wise, feedback can be collected automatically by onitoring service response times.

Automation and collaboration reduce costs by removing manual steps and standardizing ways of working. Teams can save a lot of time if they’re provided with a project template for the model and manuals that explain how to build the surrounding scaffolding for the model service.

All together, these principles help solve the three main problems identified in the introduction: production gap, long iteration cycles, and high costs.

How Is MLOps Different From DevOps?

In contrast to DevOps, MLOps recognizes that Machine Learning solutions contain not only code, but three moving parts:

Data

Used to train and evaluate models for the solution.

Model

Trained ML model used within the solution.

Code

Training code, business logic, API, or user-facing application.

Figure 2: *ML solutions consists of data, model, and code. Adapted from https://martinfowler.com/articles/cd4ml.html*

> **"Over time, a model's performance degrades, which can be due to degraded data distribution (data), due to a changed target label (model), or because the model has not been trained for a very long time (code)."**
>
> — Jasper Ginn, Data Engineer at Xebia Data.

In our use case:

To illustrate the complexity resulting from these three components, consider the classified ads use case. Assuming we already have an initial solution running in production, any change in the data, model or code will impact the behavior of the solution in the following ways:

Change in data: as the solution is adopted by more users, you notice that pictures taken in low-light conditions are not handled correctly.

- **Impact on model:** performance degrades, and the model needs to be retrained with additional low-light images.
- **Impact on code:** The app needs to be modified to warn users about low-light conditions when model predictions are uncertain.

Change in model: A new model has been developed to allow users to provide keywords for the product description.

- **Impact on data:** Keywords must be added to the dataset used to train the model.
- **Impact on code:** The app needs to ask users for optional keywords when creating a new ad.

Change in code: Developers add *cars* as a new product category.

- **Impact on data:** New training data with images and corresponding descriptions need to be gathered.
- **Impact on model:** The model needs to be retrained and evaluated with the additional dataset.

How Do I Start With MLOps?

Implementing the three MLOps principles is not just a technical exercise. Introducing new tools without having the right skills and practices in place will not improve the team's performance. Instead, building a strong MLOps capability requires investments across three different areas: people, process, and technology.

People with the right skills are key to developing ML solutions. Immature organizations often set up teams consisting of only data scientists, who focus solely on prototyping ML solutions. Mature teams have more engineering capabilities, are cross-functional and support the full lifecycle of the solution.

Processes enable rapid feedback and autonomy. Mature teams regularly get feedback from stakeholders and measure their performance. Mature organizations build platform teams in addition to ML teams. These platform teams help ML teams by providing tooling and creating common ways of working.

Technology is a driver that makes teams go faster, solutions more reliable, and projects cheaper. MLOps platforms provide a common place for teams to build prototypes and solutions. Adding testing, monitoring, and logging makes solutions more reliable. The development process can be simplified with templates, tutorials, and tooling.

The following sections dive deeper into these three pillars of people, process, and technology.

People: What Roles Do I Need for MLOps?

With companies shifting from experimenting with ML to building ML-powered solutions, the requirements for the teams have also changed. Because building production-grade solutions requires more engineering expertise, teams must become multidisciplinary and feature the right mix of business, modelling, and engineering skills.

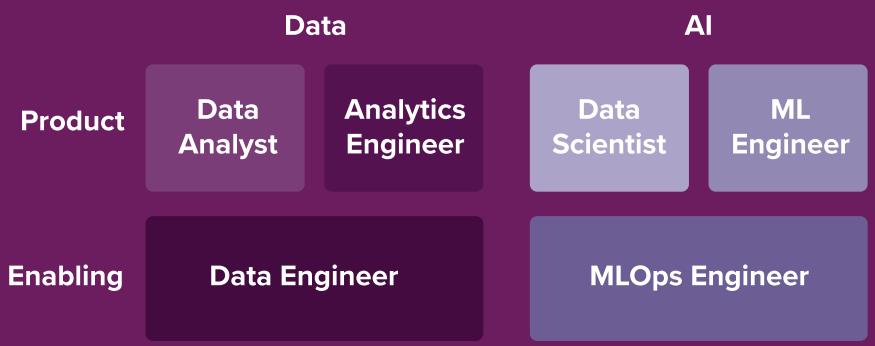

Figure 4: *Technical roles in Data and AI companies. Data scientists and machine learning engineers develop ML solutions. MLOps engineers enable ML teams by building platforms and setting out best practices.*

Teams following MLOps practices should include at least three different roles: data scientists, machine learning engineers, and MLOps engineers. We will now dive deeper into these roles, their responsibilities and skills, and how they work together on projects.

"After introducing MLOps, the organization must accept that one entire process, from start to finish, is now the responsibility of one team. That's often a huge challenge."

— Diederik Greveling, CTO at Xebia Base

Data scientists collaborate closely with stakeholders to find the shortest path from data to value. Their focus should not only be on artificial intelligence but on building the right solution, leveraging their skills in consulting and modeling.

> **In our use case:** Together with business stakeholders, data scientists from our classified ads company dive deeper into why and how better product descriptions can help users. They begin by setting up wireframes and prototypes to understand the problem-solution-fit and only then start building the model that generates product descriptions.

Machine learning engineers work together with data scientists to build robust and scalable solutions. Their engineering skills ensure that the team builds the solution right and adheres to proper engineering practices.

> **In our use case:** A machine learning engineer develops a reproducible training pipeline to automatically update the model when new product categories are added. They also embed the model into an API to connect it with other platform services.

MLOps engineers enable others to build Machine Learning solutions. They have specialized knowledge to build platforms and set out best practices that help ML practitioners scale their solutions.

> **In our use case:** An MLOps engineer adds new monitoring tooling to the platform to capture anonymized images and their generated descriptions. This data can be used to measure and improve the performance of the model.

Due to their differences in expertise, each of these roles fulfils different functions in the overall team (Figure 5). Data scientists are instrumental in using their consulting and AI skills to engage the business and build ML models, whereas machine learning engineers leverage their software and systems engineering skills to build robust solutions. MLOps engineers have similar engineering skills but focus on implementing tools and platforms that help scale the efforts of the other roles.

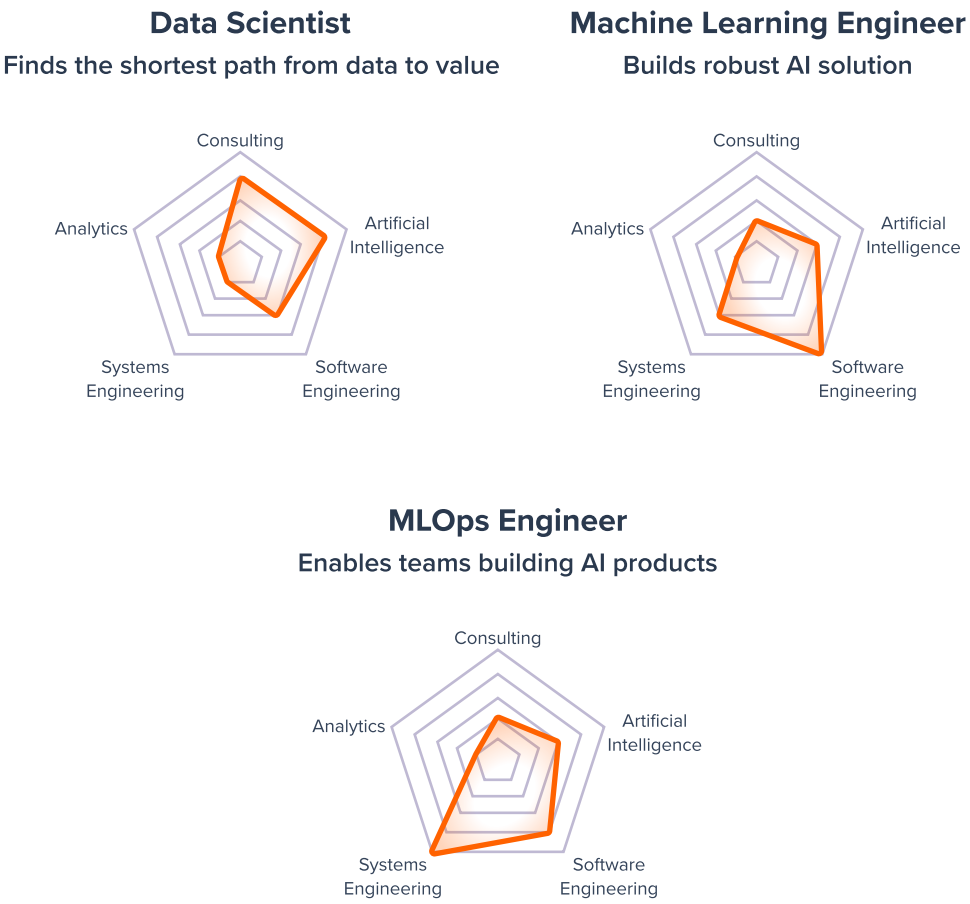

Figure 5: *Skills for technical ML roles. Data scientists need consulting and broad technical skills since they are the interface between business and engineering. Machine learning engineers combine a strong software engineering background with a keen knowledge of AI. MLOps engineers are experts in systems engineering and know how to build solid platforms.*

Similarly, due to their different focuses, these roles come into play during different parts of the process of building an ML solution. Data scientists are key at the beginning of ML projects to identify business problems and how to tackle them with ML (Figure 6). Machine learning engineers (MLES) are often involved at later stages of the process when the focus shifts to building actual prototypes and production-grade solutions.

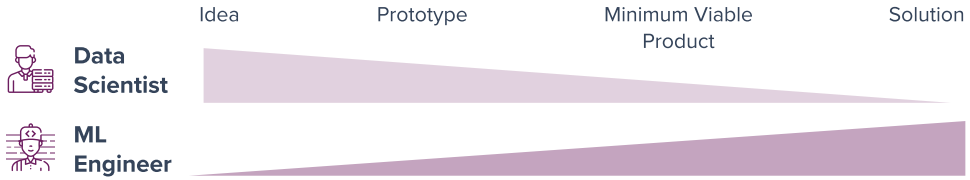

Figure 6: *Data scientists are more involved from inception to minimum viable product. Machine learning engineers turn prototypes into robust solutions.*

Understanding the differences between these three roles is key to implement MLOps in a company. Often, use-case focused machine learning engineers are hired for an MLOps role for which they don't have the passion or expertise. Similarly, hiring a machine learning engineer for a data science role can be detrimental if they are unable to engage the business effectively, while data scientists may struggle to build complex cloud solutions.

"MLOps is not so much about building a model; that's only a minor part. It's about the system around it. But eventually it's about people: having the right roles and skillsets available to tackle your challenges."

— Jasper Ginn. Data Engineer at Xebia Data.

Process: How Do I Organize My Teams to Do MLOps?

Finding the right people for a team is the first step; setting up a productive way of working is the second.

For an ML team to be efficient, it needs autonomy and full control over its solution, both during the initial development phase and when the solution is running in production. Ideally, the team should not rely on other units to build or implement its solutions, as every dependency will create delays and stall development.

> **In our use case:** The data science team builds a description generation prototype. An external engineering team spends nine months turning the prototype into a solution. Halfway through the process, the data science team comes up with a new model that uses keywords to generate better descriptions. The engineering team places this improvement on their backlog to pick it up in six months.

Figure 7: *Handovers between data science and IT teams create delays and stall development.*

"When you bring a model into production for the first time, you also want to derive a blueprint. You want to make sure teams working on similar models in the future repeat the process in a more structured and faster way."

— Roel Bertens, Principal Data Scientist at Xebia Data

In small companies (e.g. with a single ML team), the most efficient way to achieve autonomy is bringing all the required roles for building solutions into the team. This allows the data professionals to retain full control of their solution, as they have all the required expertise in house to do so.

A drawback of this approach is that it may result in large ML teams due to the number of people required to build complex solutions. Additionally, it will not scale well to larger companies with multiple ML teams, as each one of them devises their own way of working, leading to duplicated effort and reinventing the wheel.

> **In our use case:** After acquiring a machine learning engineer, the ML team can build a production-ready solution from scratch. This machine learning engineer collaborates closely with the data scientists and is also responsible for setting up the required infrastructure to run the service.

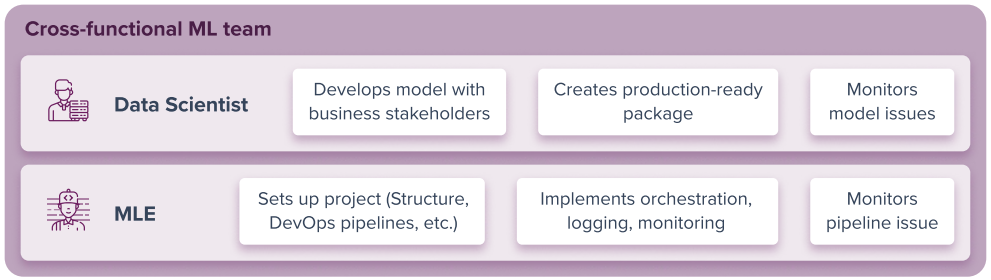

Figure 8: *Cross-functional ML teams allow data scientists and MLEs to have end-to-end ownership of their solution.*

"You want to create end-to-end responsibility, where Data Scientists don't just make the model and then pass it on. 'You build it, you run it.'"

— Rogier van der Geer, Data Charmer at Xebia Data

Introducing an MLOps platform team solves this problem by providing a shared platform and tooling to support the ML cycle. This allows ML teams to focus on building solutions rather than spending time on infrastructure maintenance, since the platform team takes care of this task. It also provides an opportunity to define common patterns, as a centralized platform team is in a better position to spot these patterns and scale them.

Figure 9: A central MLOps platform team supports multiple cross-functional ML teams by providing a shared platform and tooling.

"When introducing MLOps, you'll run into the limits of the organization long before you run into the limitations of the toolset."

— Jasper Ginn. Data Engineer at Xebia Data.

> **In our use case:** To quickly scale their item description solution, the ML team uses templates provided by the MLOps platform team to wrap their model in an API and deploy it on the shared platform. This way they don't have to build and maintain the underlying infrastructure themselves, while features such as basic monitoring metrics are provided out of the box.

An MLOps platform team should always regard ML teams as their end-users. By regularly gathering feedback and engaging with the ML teams, the platform team focuses on building tooling that their users want and need.

While platform teams help scale tooling, ML communities help scale knowledge across teams. A well-connected community plays a crucial role in sharing common practices and knowledge so that teams can learn from each other and up their game. Regular meetings and workshops support this mission, but so do shared standards, tutorials, and project templates.

"Very often both the users and the data scientists don't know what's possible to develop, because they lack the other side's perspective. That's why communication and iteration are needed to improve the solutions."

— Joost Bosman, Machine Learning Engineer at Xebia Data.

Tech: Why Do I Need an MLOps platform?

An MLOps platform provides data scientists and machine learning engineers with the infrastructure and tooling to build and deploy reliable ML solutions.

MLOps platforms are often depicted as complex structures with many components, but these can be boiled down to the following:

- **Workspaces** allow users to interactively experiment with datasets and predictive models.
- **Model pipelines** take care of preparing datasets and training machine learning models.
- **Model registries** are used to store machine learning models and their metadata.
- **Model deployment** supplies standard ways to serve predictions.
- **Model monitoring** tracks performance of the model when generating predictions.

Data scientists and machine learning engineers interact with the ML platform through the workspace, where they can develop code to explore data and train models using notebooks or an integrated development environment (IDE). From the workspace, they can also create pipelines to automatically train models, thus improving reproducibility. Trained models are stored in the model registry to keep track of different model versions over time and their performance.

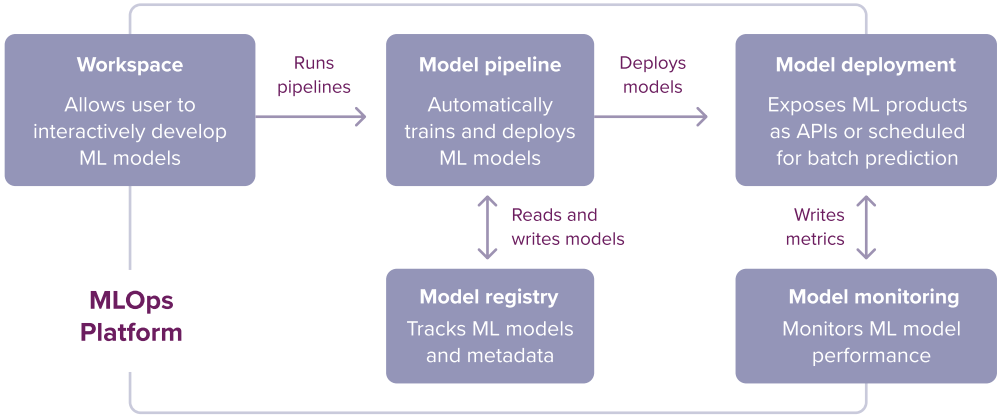

Figure 10: *Essential components of an MLOps platform.*

Depending on the use case, models stored in the registry can either be deployed for batch predictions or online predictions. In the case of batch predictions, models are loaded into pipelines that generate predictions on a recurring schedule (e.g., daily, weekly, etc.). For on-demand predictions, models are packaged into APIs and deployed as microservices that can be integrated with other (user-facing) applications.

> **In our use case:** To integrate our item description model with the app, a machine learning engineer wraps the model in an API and deploys it as a microservice that can be called by the application front-end. To allow automatic retraining of the model, he also creates a model training pipeline that creates a new version of the model whenever the training dataset changes. The result of this pipeline is stored in the model registry and deployed if performance is satisfactory.

MLOps platforms equip ML product teams with the infrastructure and tooling required to build and deploy ML products. As for all platforms, it is key that the platform closely mirrors the ways of working of the teams. But to truly scale MLOps within organizations, teams need one additional component: a Golden Path.

What Makes an MLOps Platform Different to Other Platforms?

MLOps platforms are clearly distinguishable from other platforms because of their focus on machine learning.

For example, compared to an application platform, ML projects need complex orchestration pipelines and larger hardware requirements than many non-ML applications. Compared to data and analytics platforms, ML projects require workspaces to interactively develop Python code with ML frameworks, a model registry to track different versions of trained models, and the capability to expose models as APIs for on-demand predictions.

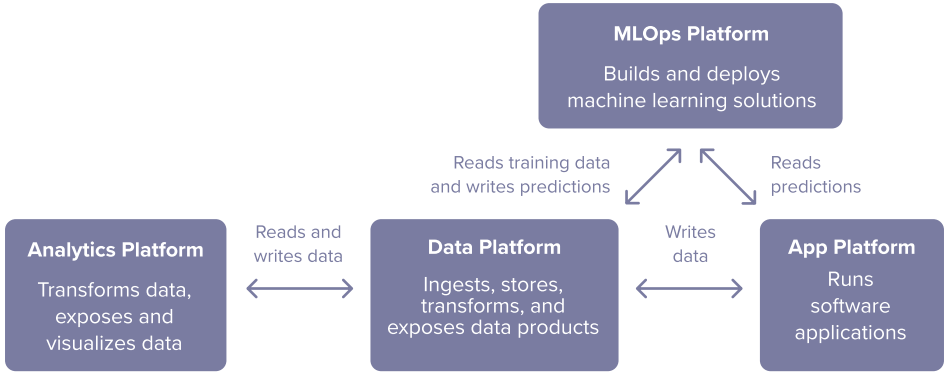

Figure 11: *Relation between App, Data, Analytics, and MLOps Platforms*

Good integration between these platforms is key to unlocking their synergies. If the data platform provides frictionless access to datasets, the MLOps platform can focus on capabilities specific to machine learning. Similarly, dashboarding should be part of an analytics platform, while user-facing applications should run on an app platform. Each platform has its own focus.

> **In our use case:** In our Classified Ads Platform, the front-end app for end-users runs on the application platform. Whenever a user uploads a photo for a new ad in the app, this app calls the description generation service on the MLOps platform. The model inside this service was trained using a historical dataset of pictures and descriptions stored in the data platform. The performance of the model and app services are visualised on the analytics platform.

Platform	Users	Artifacts	Storage	Example tooling
Data	Any data consumer	Data	Code, data	SQL, Spark, Fivetran, Databricks
Analytics	Data analysts, analytics engineers	Data, analyses, dashboards	Code, data	SQL, dbt, PowerBI, Tableau, SnowFlake
MLOps	Data scientists, machine learning engineers	Data, model, pipeline, API	Code, data, model	Python, PyTorch, Jupyter Notebook, MLFlow
App	Software engineers	API	Code, data	Java, Grafana, databases

Table 1: *Illustrations of how MLOps platforms are different from data, analytics, and app platforms.*

Tech: How Do I Quickly Build Robust Solutions?

An MLOps platform is a key enabler for ML teams, but it is not enough. Without a common way of working, teams spend too much time reinventing the wheel and their collaboration is limited. A Golden Path guides users on how to tackle common use cases with the tooling of the platform.

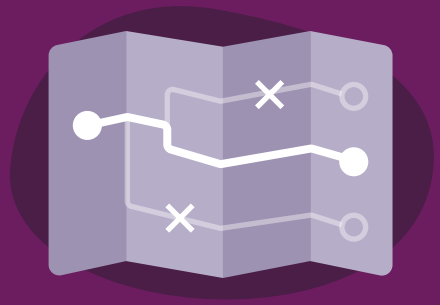

Figure 12: *A Golden Path represents an easy but opinionated way of working. It helps users avoid common pitfalls.*

Golden Paths represent an easy but opinionated way of working. Each guide consists of a tutorial supported by specific tooling and project templates to help bootstrap new projects. This guidance allows users to roll out new products with minimal effort.

"Our Golden Path helps users train and deploy new ML models on our ML platform. The path consolidates the best parts of existing practices and provides a clear recommended way of working supported by guides, templates, and shared components. This allows our teams to deploys models in days rather than months."

— Julian de Ruiter, Principal Machine Learning Engineer at Xebia

Golden Path	Model training	Model deployment	App development
Objective/ use case	Train a machine learning model and log it in a model registry	Deploy a model from the model registry as an API	Build a user-facing iPhone application
Audience	Data scientists	Machine Learning Engineer	Software/UX Engineer
Tooling	Python, PyTorch, MLflow, Kubeflow pipelines	Python, GitHub Actions, Docker, Kubernetes	Swift, iOS, xCode
Materials	• Tutorial on using the model registry • Guidelines on building ML pipelines • Project template	• Tutorial on containerizing and deploying models • Guidelines for monitoring deployed APIs • Project template	• Tutorial on how to build a basic iOS app • Guidelines for best practices • Project template • Branding materials

Table 2: *Example Golden Paths for ML projects.*

Golden Paths are powerful because they strike a healthy balance between development speed and quality. They do not dictate what users should use but represent a recommended approach. Users are allowed to follow other paths if needed but can always rely on this kick-starter for their projects. Golden Paths represent a pragmatic middle ground between a developer free-for-all and overly restrictive IT-enforced platforms.

Platform	Developer free-for-all	Golden path	IT-enforced platform
Experience	No guides, little documentation, no/poor standards	Guides for common use cases, recommended tooling, quality standards	Focus on ops, very strict standards, overly constrained
Tooling	Many different tools across projects	Supported toolchain with options for customisation	One tool to rule them all
Speed	●●●	●●	●
Quality	●	●●	●●●

Table 3: *Golden Paths are a pragmatic middle ground between a developer free-for-all and an overly restrictive IT-enforced platform. Adapted from https://thenewstack.io/the-cloud-native-paved-path-developer-experience.*

What Makes a Good Golden Path?

Successful Golden Paths are generally not dictated by a central team but developed in collaboration with the envisaged users. A good approach is to look at existing practices within the company and see how they can be generalized and scaled across teams.

Good Golden Paths have a clearly defined audience and keep their knowledge and experience in mind. Similarly, they have one main purpose to avoid becoming too bloated. However, they should also not be made any shorter than necessary as they should not omit any key details.

For wide adoption, it is also important to make Golden Paths a key part of the onboarding and education of new teams or hires. This will help people come onboard the platform and learn the standards/tooling recommended within the company.

Finally, teams should always test and gather feedback on the Golden Paths – the only thing worse than having no Golden Path is having a wrong one.

"The MLOps landscape is still very technical, but it shouldn't be. There's a lot of need for abstracting the details away. You should be able to do MLOps with a good understanding of Data Science and not have to learn Kubernetes just to train and deploy a model."

— Julian de Ruiter, Principal Machine Learning Engineer at Xebia

Bringing It All Together: How Can I Measure My Progress?

As discussed in this whitepaper, MLOps encompasses people, process, and technology.

Mature MLOps organizations use cross-functional ML teams with well-defined roles, such as data scientists, machine learning engineers, and MLOps engineers. These teams are autonomous and responsible for the solution from inception to production.

Regular feedback from end users is essential for success. Short iteration cycles allow for quick improvements. Dedicated MLOps teams help improve efficiency and autonomy by building MLOps platforms and setting out best practices.
See Table 4 for a complete comparison between Immature and Mature stages.

	Immature stage	Mature stage
People	• Focus on hiring mostly Data Science roles • Little engineering skills within teams • Practitioners have narrow view of their role	• Cross-functional teams with well-defined roles (DS, MLE, MLOps Engineer) • Teams have skills for end-to-end responsibility • Senior in-house talent
Process	• Strong dependencies between teams • Stage gates cause long delays • Unstructured or ad hoc knowledge sharing • Approval needed for changes	• Regular feedback from end users • Short iteration cycles • MLOps platform teams • ML Team can work autonomously • Learning is part of the culture
Tech	• Teams need to reinvent the wheel to deploy their solutions • Solutions are not reliable • No platform in place • Little automation, lots of manual steps	• Platforms to scale experimentation and production • Golden Paths to make it easier to bootstrap projects • CI/CD for model, data, and code

Table 4: *How to mature your MLOps Capability: examples of maturity indicators.*

These indicators measure the state of the overall MLOps capability, but how to evaluate the performance of teams across this journey?

The performance of ML teams can be measured with the four software delivery performance metrics defined by DORA (DevOps Research and Assessments, see Table 5). The faster you can make changes to your ML models ('deployment frequency' and 'lead time for changes' metrics), the sooner you can deliver value to your users and gather their feedback. The more reliable your solutions are ('change failure rate' and 'time to restoration' metrics), the happier your users will be and the more time your team will have to work on improving models rather than just maintaining them.

ML teams – DORA metrics	MLOps team - Platform metrics
1. **Deployment frequency**. How does a team release changes to an ML model in production?	1. **Platform adoption**. What percentage of ML teams have become platform users?
2. **Lead time for changes**. How long does it take for changes to get deployed to production?	2. **User happiness**. Are ML teams happy with the platform services offered?
3. **Change failure rate**. How many deployments cause a failure in production and require intervention?	3. **Capabilities coverage**. What capabilities are supported by the platform?
4. **Time to restoration**. How long does it take to recover from a failure in a production model?	4. **Cost per user**. What is the cost per user, including development and licensing costs?

Table 5: *Performance metrics for ML and MLOps teams. Adapted from https://dorametrics.org/*

MLOps platform teams should focus on ML teams, but their success should not only be measured with the metrics of ML teams. Additional product metrics should show how many teams are using the platform ('platform adoption'), how satisfied those users are ('user happiness'), how well it meets their needs ('capabilities coverage'), and how cost-effective the platform is ('cost per user').

For data-driven teams, it is ironic that we often don't measure their performance. Using the indicators and metrics from Table 5 will make your MLOps practices data-driven.

How Do I Assess My AI Maturity?

Our whitepaper "The AI Maturity Journey" describes the phases that companies go through when becoming AI-driven. Read the whitepaper for the full picture!

Meet the Experts

Henk Griffioen – Principal Data Scientist at Xebia Data

Henk builds AI products and AI teams. He's a data scientist with engineering chops and a background in machine learning. Whether it's tech or business, he loves simple and clear solutions.

Julian de Ruiter - Principal Machine Learning Engineer at Xebia Data

Julian is an avid tech enthusiast who enjoys trying new technologies and engineering practices and applying these to the realm of Machine Learning and AI. He also wrote the Manning book 'Data Pipelines with Apache Airflow', which aims to bring aspiring users quickly up to speed with Airflow.

Diederik Greveling – CTO at Xebia Base

Diederik delivers easy-to-maintain data and cloud-oriented products that quickly create value. He has an engineering background and focuses on distributed systems, cloud computing, and data architecture design.

Jasper Ginn – Principal Data Engineer at Xebia Data

Jasper is a data & ML engineer and has previous experience as a data analyst and data scientist. He deeply believes in life-long learning and is always seeking to expand his knowledge.

Travis Dent - Data/ML Engineer at Xebia Data

Travis builds robust systems following solid engineering principals and striving for rapid value delivery. He is experienced in gathering requirements, architecting solutions and leading solutions teams.

Daniel van der Ende - Principal Data Engineer at Xebia Data

Daniel enjoys working on high performance distributed computation with Spark, empowering data scientists by helping them to run their models on very large datasets with high performance. He is an Apache Spark and Apache Airflow contributor and speaker at conferences and meetups.

Rogier van der Geer – Data Charmer at Xebia Data

Rogier is an experienced data scientist and machine learning engineer with a PhD in particle physics. With a passion for applying data science to solve real-world problems, he is driven by the desire to make a positive impact on society through his work.

Joost Bosman - Machine Learning Engineer at Xebia Data

Joost is a machine learning engineer with a research background and a passion for technology. His curiosity and creativity encourage him to explore and come up with out of the box solutions with a healthy dose of pragmatism.

Roel Bertens - Principal Data Scientist at Xebia Data

Roel is helping customers by building data products so they can enjoy the value of their data. He is using both his scientific skills as well as his computer programming background to deliver high quality software.

Jetze Schuurmans - Machine Learning Engineer at Xebia Data

Jetze is a well-rounded Machine Learning Engineer, who solves Data Science use cases and productionizes them in the cloud. His interests include MLOps, GenAI, and Cloud Engineering and loves sharing knowledge during community events or by giving training.

About Xebia

Xebia is a trusted advisor in the modern era of digital transformation, serving hundreds of leading brands worldwide with end-to-end IT solutions. The company has experts specializing in technology consulting, software engineering, AI, digital products and platforms, data, cloud, intelligent automation, agile transformation, and industry digitization.

In addition to providing high-quality digital consulting and state-of-the-art software development, Xebia has a host of standardized solutions that substantially reduce the time-to-market for businesses. Xebia also offers a diverse portfolio of training courses to help support forward-thinking organizations as they look to upskill and educate their workforce to capitalize on the latest digital capabilities.

The company has a strong presence across 16 countries with development centers across the US, Latin America, Western Europe, Poland, the Nordics, the Middle East, and Asia Pacific.

Interested in bringing MLOps into your organization? Please contact:

Julian de Ruiter,

Principal Machine Learning Engineer

Julian.DeRuiter@xebia.com