**Lightweight Introduction to MLOps**

**How and where the MLOps journey starts — basic building blocks**

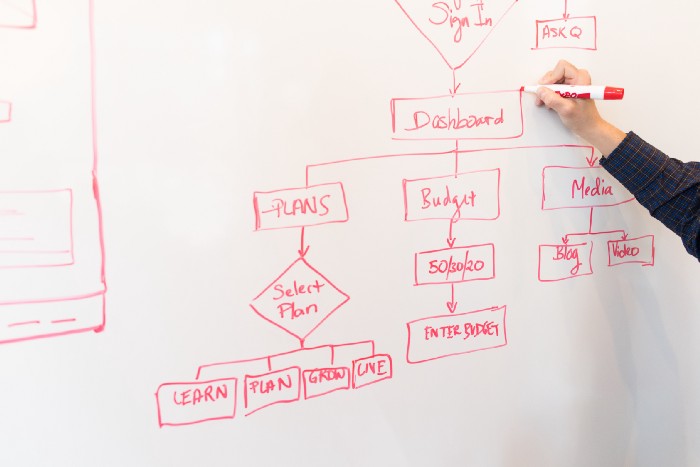


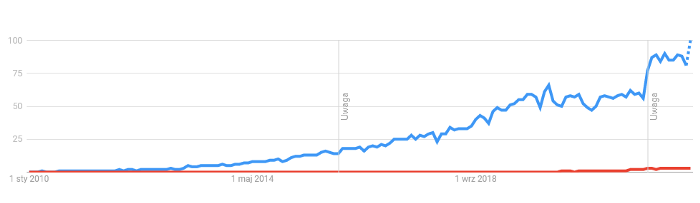
Photo by [Christina @ wocintechchat.com](https://unsplash.com/@wocintechchat?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com?utm_source=medium&utm_medium=referral)

1. **Introduction**

You may have heard that [90% of ML models don’t get into production](https://towardsdatascience.com/why-90-percent-of-all-machine-learning-models-never-make-it-into-production-ce7e250d5a4a). Actually, any IT practitioner knows that putting any software into production is a long and complex way and a challenge in itself. However, over many years, since people started to write first if-clauses, constant improvement in processes, ways of development, deployment and servicing was present. This led to the establishment of so-called DevOps processes and tools. Nowadays, these are incorporated almost in every company creating serious software, doesn’t matter if in gaming, manufacturing, banking or the medical industry. There are now hundreds if not thousands of web pages and articles written about this topic.

However, in recent years a new set of software types emerged in the world’s light, namely AI-based systems. They use a significantly different approach to solving problems — they are based on statistics, probability and, most importantly, a lot of data. This creates a new set of challenges that cannot be effectively tackled with standard DevOps methodologies (as processes are somehow different). Many companies who tried this, failed.

Because this is a much more complex and challenging field and a new specialisation was recognised in the IT world — MLOps. Unfortunately, this is still a very young profession — it can be easily seen by checking a popularity of phrases “MLOps” and “DevOps” in Google trends. Since about 2019 it was essentially not existing.



Blue — DevOps, Red — MLOps; image by author from Google Trends

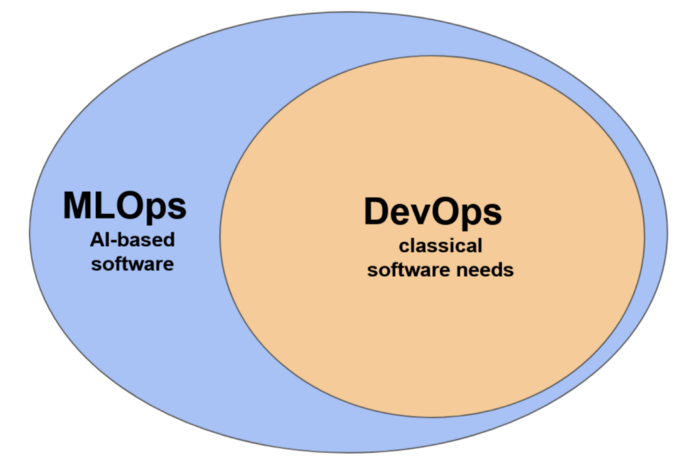
Because of that, there are not so many definitions, rigid rules or proven methodologies that can be easily adopted. Every AI-based company still experiments and strives to find the best way to approach the problem of the effective creation and deployment of AI systems. However, if you like definitions here’s one you can find on [Google Cloud Platform’s site dedicated to MLOps](https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning):

*MLOps* is an ML engineering culture and practice that aims at unifying ML system development (Dev) and ML system operation (Ops).

So it’s not only about faster model development — it’s about the big picture and optimising processes. That’s the reason why this article was written — to give you a light, informal introduction to the field of MLOps. It’s in no way restrictive — it highlights my personal experience, backed with some research about this topic and should be treated as food for thought for anyone considering the introduction of MLOps in their company or entering the MLOps field as a professional.

**2. MLOps challenges**

First, what are these challenges which make MLOps distinct from DevOps? You should think of MLOps as an extended version of DevOps as it tackles the same problems plus some additional ones.



Scopes of MLOps vs. DevOps; image by author

Let’s look at some of the challenges:

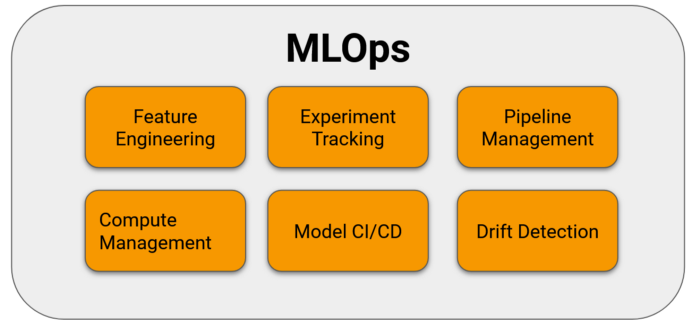
1. First, the ML models heavily rely on statistics and probability. Their internal parameters are not set directly by developers (called ML engineers or data scientists) but indirectly by setting the so-called hyperparameters controlling the behaviour of the algorithm.
2. The input to the system is flexible and not controlled. By this, I mean that the internal behaviour of the system is optimised based on the historical data (ML/Dev phase) but after deployment, it acts on the real-time data (Ops phase). If, for example, the client’s behaviour changes, the system will still follow the old decision patterns which it learned based on old data. This will lead to a rapid depreciation of its value. The process behind it is called data drift and is one of the biggest challenges in the Ops phase of AI-based systems. Let’s take an example — a pizza ordering system where you ordered a pizza pepperoni and received a Hawaiian pizza — you can easily map and track down the issue in your code and fix it. In ML systems this cannot be done so easily.
3. Another challenge often comes from the background and education of developers. Because ML systems are based on things like for example highly advanced linear algebra, Bayesian statistics and probabilistics, the ML specialists have a different focus during their education. Compare that with for example typical frontend engineers (how often do they have to use e.g. a matrix decomposition?). In practice, this means that during development, they often use frameworks that hide some complex software-related details, in favour of ease of use (e.g. Keras, sklearn). These frameworks and libraries are under constant development and changes or an emergence of a new version of ML algorithm is quite likely (e.g. development in the transformers domain). In short ML engineers:

* do not always have complete, fine control of the algorithms they use
* like and have to experiment with new algorithms and methods
* are much more on the math side than classical software developers

As you see, overall the issue is in the flexibility of both data and algorithms. This is the biggest advantage of it but also the biggest downside.

**3. Implementation of MLOps**

MLOps aims to take under control both development and in-production issues in an organised way. To achieve that, there are some important functional building blocks to be adopted — as shown in the picture below. There may be more depending on the specifics of the industry or company but these are usually common across various use cases.



Basic, top-level components of MLOps; image by author

Let’s talk about each of them shortly.

1. Feature engineering is about automation of ETL pipelines and their version control. Ideally you would have something is a style of a feature store. If you are not familiar with this concept check this [website](https://www.featurestore.org/). Some tools available on the market: Databricks Feature Store, Feast, Tecton.
2. Experiment tracking is a big and really crucial component because it is dealing with both ML engineers experiments — both those successful and failed too. It allows to revisit some previous ideas (like different algorithms or a features) when the time comes, without reinventing a wheel. In a mature ML system there is also a way to capture set of hyperparameters (past and current) and corresponding system quality KPIs — usually this is called a model registry (tools like MLflow, Neptune or Weight&Biases).
3. Pipeline management allows you to version control the pipeline which controls flow of data from the input to the output. It should also log each run and rise a meaningful error if something bad happens. Here take a look at: Vertex AI Pipelines, Kedro, PipelineX, Apache Airflow.
4. Compute management tackles the problem of scalability in ML systems. Some algorithms require tremendous amount of computational power while training and retraining and little during the inference. As often these two tasks are connected by a feedback control loop the system must be able to scale up and down. Sometimes additional resources like GPU must be attached for training while not required for the inference part. Public cloud providers nicely tackle this issue offering autoscaling and load balancing.
5. Model CI/CD is very similar to the CI/CD from the DevOps area but additional checks have to be performed before deployment of the model. These are chosen performance metrics that have to be in an acceptable range and always compared with the current model in production. One of the most popular tools here are Jenkins and Travis but there is plenty of others like TeamCity or Circle CI.
6. Drift detection is a module that monitors the characteristics of the incoming data and behaviour of the system. When the characteristics of incoming data deviate from the expected range an appropriate alert should be rised so retraining of the model can be requested (automatically or manually). If this doesn’t help, the alert should be escalated and the dev team should take a deeper look into the issue. Tools/services to consider: AWS SageMaker Model Monitor, Arize, Evidently AI.



Exemplary tools for MLOps environment; image by author

When introducing MLOps to your organisation, especially if it was, or still develops software, you must be very careful and aware of the DevOps-bias. Many people not familiar with the AI domain will be inclined towards proven DevOps solutions — and will be pushing for them. That’s why in many companies ML-based systems are being developed by a dedicated teams or departments.

**4. Summary**

As you see this article was introductory and generic not to impose any specific solutions for you. That is because various companies have its own internal processes to be automated and specific challenges that may require different approaches and tools to find the most optimal solution. Still, there are some great materials dealing with MLOps that are written with a specific set of tools in mind. that I highly recommend, for example:

* <https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning>
* <https://azure.microsoft.com/en-us/products/machine-learning/mlops/#features>
* <https://aws.amazon.com/sagemaker/mlops/?sagemaker-data-wrangler-whats-new.sort-by=item.additionalFields.postDateTime&sagemaker-data-wrangler-whats-new.sort-order=desc>