GUIDE TO

Evaluating MLOps Platforms

November 2021

/thoughtworks

Table of contents

Introduction

The MLOps scene is exploding, with a multitude of tools to consider. It's what analyst house Gartner calls a "glut of innovation". The rapid expansion of MLOps platforms is only one part of the overall AI landscape and just the number of MLOps platforms alone has grown rapidly.

This presents a big challenge for those looking to use these platforms: how to decide between them? This guide gives you a jumpstart towards answering that question for your organization.

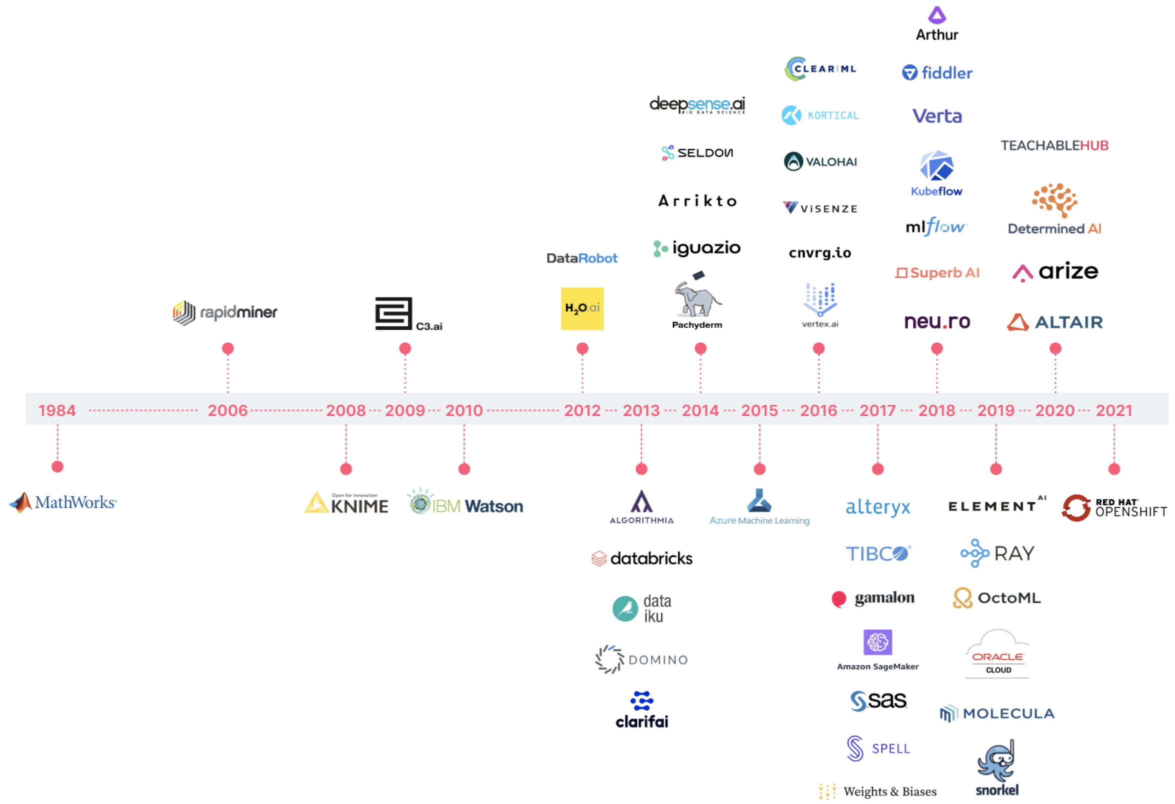


Figure 1: MLOps Platform timeline[1]

[1] For big companies we have given the year that they entered the MLOps market. Google has only used the name Vertex since 2021 but we show it here in the year that Cloud Machine Learning Engine was launched. For startups we have used founding dates. MathWorks is an outlier because MATLAB really did start in 1984.

Purpose of this guide

In this guide we will support users and buyers by showing:

1. How to make sense of the MLOps platforms space
2. How to pick the right platform evaluation criteria for your organization (no one-size-fits-all)
3. How to structure an MLOps platform evaluation
4. how to use the open-source material we provide to get a bootstrap on your evaluation

Viewed from a distance, end-to-end MLOps platforms appear to have the same goal of supporting the ML lifecycle. But when you get into the details, these platforms have different goals; direct comparisons of platforms against each other is fraught with difficulty.

Not every vendor has the same picture of how an MLOps platform relates to the ML lifecycle or exactly what purposes should be served. These differences reflect the different needs organizations have from an MLOps platform. There is no one-size-fits-all.

We are not trying to find 'the best MLOps platform'. We want to help organizations buy/build/assemble an MLOps platform that can best serve their particular needs.

We recommend a phased approach to evaluation with a narrowing list of candidate platforms. Initial candidates are chosen based on high-level criteria, such as fit for the organization's cloud/technology strategy, alignment to other tools in the tech stack and support for key use cases. A detailed evaluation then goes to the level of features and PoCs. We explain how to construct comparisons for the detailed phase later in the guide.

We will first understand how platforms relate to the ML lifecycle and how this opens up space for different platforms to take different approaches. We will then consider pitfalls in choosing evaluation criteria. We then explain how to structure an evaluation and how to use the open-source material we provide to get a bootstrap on your evaluation. Finally, we will analyze trends to be aware of in the MLOps field.

Intended audience

The primary audience for this guide is the owner of an MLOps platform initiative — somebody tasked with bringing an MLOps platform into the organization. Anybody interested in MLOps platforms can also benefit from this guide. We do assume some knowledge of the challenges of the field. For more background on the challenges of MLOps see our previous ebook How to get MLOps right or our article defining the concept of Continuous Delivery for Machine Learning (CD4ML).

Purpose and variation of MLOps platforms

The purpose of an MLOps platform is to automate tasks involved in building ML-enabled systems and to make it easier to get value from ML. There are many steps involved in building ML models and getting value from them: steps such as exploring and cleaning the data, executing a long-running training process and deploying and monitoring a model. An MLOps platform can be thought of as a collection of tools for achieving the tasks involved in getting value from ML. But a good platform is not only a collection of tools, the tools should fit together into a shared approach that provides consistency, both end-to-end consistency in how activities are handled and also consistency across the organization as a single platform for different projects and departments.

In this section, we'll explore the details of what the purpose of MLOps platforms is. We'll also understand why there's so much variation in MLOps platforms and we'll use this to better understand the landscape of MLOps platforms.

MLOps platforms and the ML lifecycle

To get a more detailed picture of what MLOps platforms do, we need to understand the ML lifecycle — the process of building and getting value from ML. If there were a common understanding of the ML lifecycle, MLOps platforms would be easier to understand than they currently are. Unfortunately, the industry hasn't agreed on the details. If you search for 'ML lifecycle', you'll find diagrams that only roughly agree on where the lifecycle begins and ends and what activities fall under it.

Here is an end-to-end picture of the ML lifecycle:

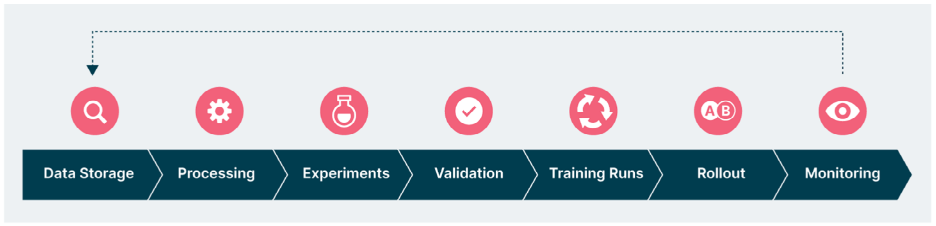


Figure 2: ML lifecycle

Notice this lifecycle starts with discovery activities based on data — finding data, exploring it and understanding the problem. The lifecycle here ends with monitoring — observing the performance of the model in live and gathering data to assess performance. The arrow back to the beginning represents improvements that can be made in the light of performance data.

A big source of variation in depictions of the ML lifecycle is whether discovery and monitoring are included and whether there is a feedback loop. If you look deep enough into ML lifecycle depictions, you also find differences in which activities fall under the lifecycle (e.g. is data or model security included?) and who is responsible for them (e.g. do data scientists deploy to production?). Each platform takes its own view on open questions about who plays which roles in the ML lifecycle and which activities are part of the ML lifecycle. When activities such as data discovery or monitoring aren't seen as part of the ML lifecycle, they're seen as external tasks — falling under data tooling rather than ML tooling or a task for operations rather than any ML-specific role.

It's important to emphasize that the ML lifecycle diagram begins with discovery activities and includes a feedback loop from monitoring. Not every ML project will require this but too many projects underestimate the need to handle on-going change and management of data[2]. If this is not all handled by an MLOps platform itself, then it can be handled with integrations to other tools and platforms. An evaluation process needs to identify these points of integration.

[2] We have emphasized this in previous Thoughtworks publications such as Continuous Delivery for Machine Learning and How to get MLops right.

Overlaps with non-ML tools and platforms

We've seen already that varied understandings of the ML lifecycle lead to variation among MLOps platforms. Further complication is added by overlaps with other functions/platforms in the organization, especially data platforms.

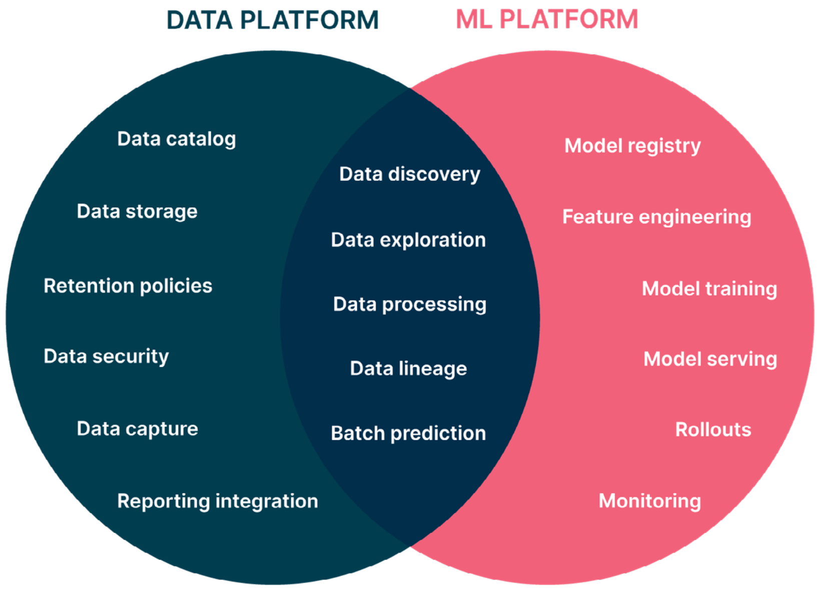


Figure 3: Overlaps between data and ML platform

Some vendors pitch their data platform and MLOps platforms as seamlessly integrated. Databricks, for example, has MLflow aimed at MLOps; it pitches MLflow as part of its Lakehouse Unified Data Analytics Platform. Other vendors emphasize data and MLOps separately and highlight points of integration. For example, Microsoft Azure has a tool called Azure Synapse Analytics, which can be used with Apache Spark for data transformations. This is a data platform feature and, for data wrangling, can integrate with an Azure Machine Learning workspace.

For organizations that already have a data platform, overlaps lead to questions about what parts of an MLOps platform are needed by the organization. Tools used for data processing and ETL, such as Airflow, can also be used for automating training and delivering machine learning models. Some organizations might therefore find it sufficient to train models with Airflow. If such an organization also has adequate ways of deploying and monitoring ML models then they may not need to introduce an end-to-end MLOps platform.

Here are other areas of overlap between Data and MLOps platforms that can lead to questions about how much of an MLOps platform is needed and where points of integration may be needed:

- Data lineage is a concern for jobs such as cleaning ML data or training ML models, as well as ETL operations within data platforms. Delta lake, for example, comes at this from a data platform perspective (though can be used for ML) and Pachyderm comes at this from an MLOps perspective (though can be used for ETL)
- Obtaining batches of predictions from machine learning models can be achieved with a processing engine like Spark or the predictions can be made and stored to a file in an ETL tool like Airflow. Alternatively, a feature within an MLOps platform can be used, either as part of model serving or part of an ML pipeline
- Feature stores capture features that have been engineered from the raw data with cataloging for reuse. The feature store therefore offers a more processed form of the data in the data platform, similar to the role of a data mart alongside a data lake (though a mart typically serves reporting rather than ML)

Overlaps with data platforms and data tools pose questions at the data storage, processing and experimentation phases of the lifecycle — the earlier stages.

There are other overlaps with analytics, monitoring and reporting tools that lead to integration questions at the monitoring stage. At the monitoring stage we find these patterns in MLOps platforms:

1. Basic information about the number and rate of predictions served and how much resource this is using may be exposed by default
2. Some ML-specific monitoring is done in a way that only requires minimal coding/configuration e.g. some approaches to drift detection
3. Some monitoring is specific to the use case and model. For these cases we find vendors offering export of data for integration into other tools

Its important to be aware of this when evaluating MLOps platforms. If export or integration to another tool is needed (point 3) then you need to be sure you have a suitable tool or can use one that the vendor suggests. You also may already have tooling for the kind of basic monitoring covered in point 1 (e.g. Prometheus, Splunk or similar). An MLOps platform may not integrate with your established tool, so you may then have a question about whether your organization is happy to do monitoring in a different tool for machine learning.

Who performs which MLOps tasks?

Another axis of variation is who the MLOps platform serves. In some organizations one might find work on ML products divided as follows:

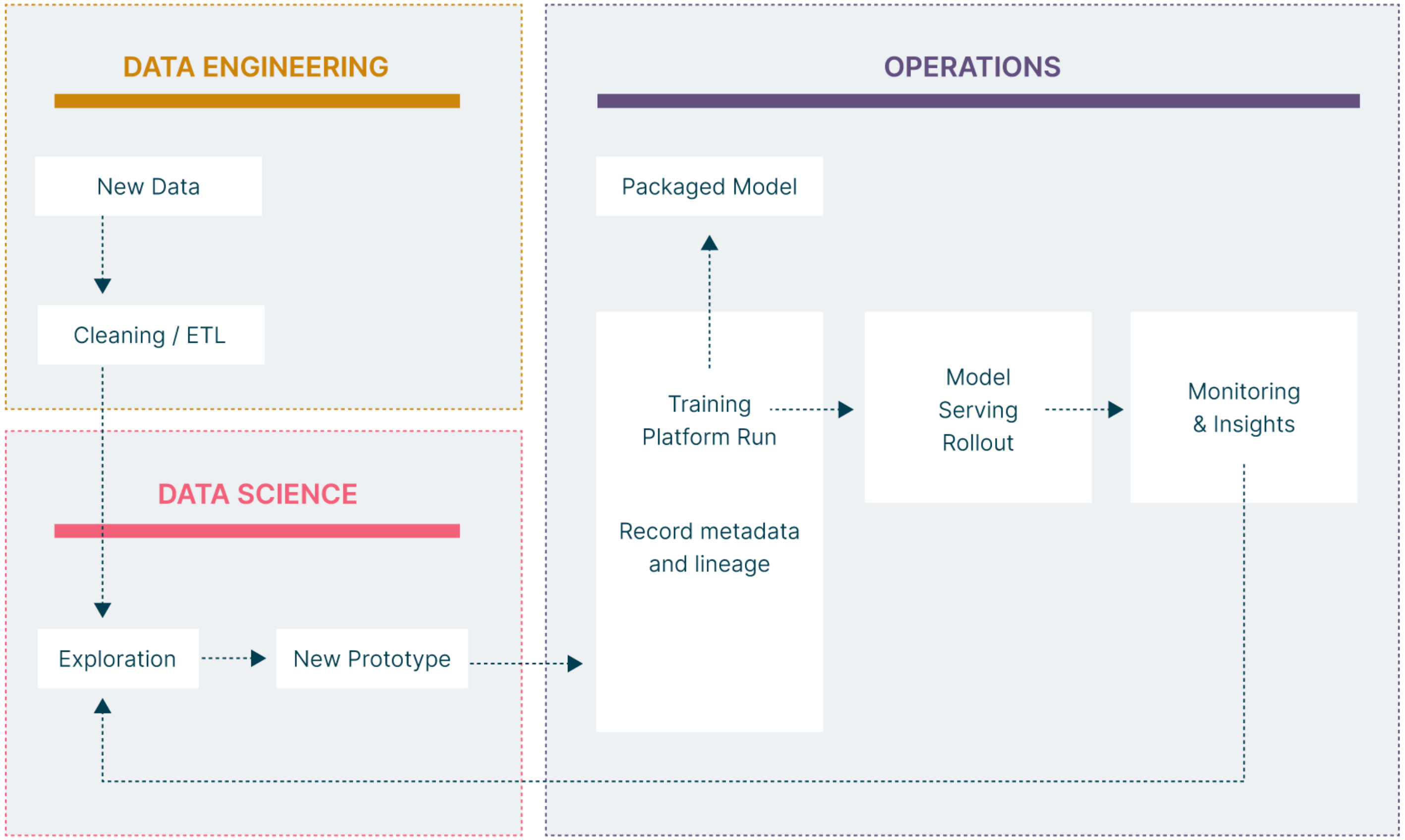


Figure 4: Division of work on ML products

This would map to:

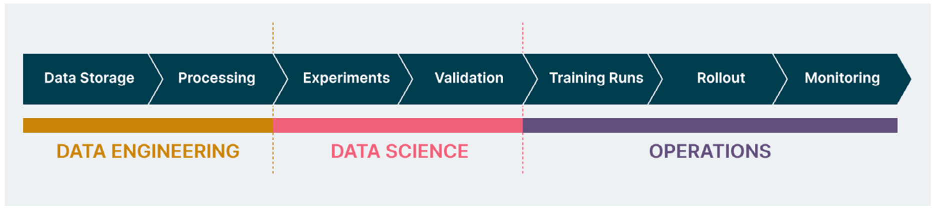


Figure 5: Division of work on ML products mapped to lifecycle

But what about organizations that have machine learning engineers?[3] Do the role boundaries then become more fluid, with machine learning engineers taking over from both operations and data engineering?

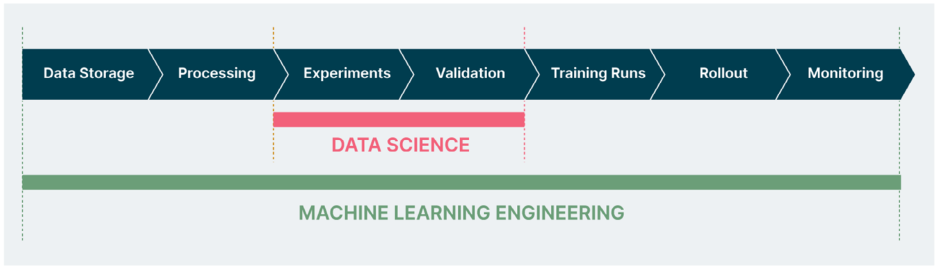


Figure 6: Division of work on ML products mapped to lifecycle — with machine learning engineers

Further, some platforms aspire to make more of the lifecycle accessible to data scientists and some data scientists possess operations skills.

The boundaries of roles vary from one organization to another. This is important when assessing MLOps platforms, as some platforms are opinionated about which lifecycle activities are owned by which users — and how permissions are granted. If you choose an opinionated platform then you'll want it to be aligned with your organization and the goals of your platform initiative. For example, if you only want your operations team to be able to deploy models to production then you'll want to look at how you can achieve that with the platform.

The most radical variation in terms of target personas comes with whether and how MLOps platforms incorporate automated machine learning (AutoML). When AutoML is the core focus of the platform, the need for deep data science knowledge and operations knowledge are both reduced; instead, the key role is a 'citizen data scientist' or even an advanced business user. This can mean the screens avoid complex configuration options (or hide those requiring deep knowledge). We'll return to this tension later in the section Approaches and target personas — trade-offs of automation'. For now it's enough to note that there are open questions in MLOps about which personas perform which tasks — and that this is reflected in variation in MLOps platforms.

[3] Here we are sidestepping the question of what roles are optimal. For our purposes here we assume that readers must choose a platform to fit the organization.

Making sense of the landscape

In this section, we will provide some conceptual tools for categorizing software in the MLOps field. Our purpose is not to provide definitive categorizations, but to **provide categories to help organizations filter the hundreds of MLOps tools down to the most relevant.**

The most important category is the distinction between specialist tools aimed at a particular purpose and end-to-end platforms that support the whole lifecycle. Specialist tools target particular activities in the ML lifecycle, for instance data wrangling.[4] End-to-end platforms provide functions that overlap specialist tools — they can be thought of as assemblies of specialist tools that have been designed to work together. One challenge of navigating the MLOps space is that there is no clear line between tools and platforms.

[4] Specialist tools are also referred to as 'best of breed' tools.

Understanding tools vs platforms

It can be helpful to think of specialist MLOps tools and end-to-end MLOps platforms as falling on a spectrum of specialisation versus generality. Some specialize in a specific activity in the lifecycle, some specialize in multiple parts of the lifecycle and some (platforms) aim to cover everything.

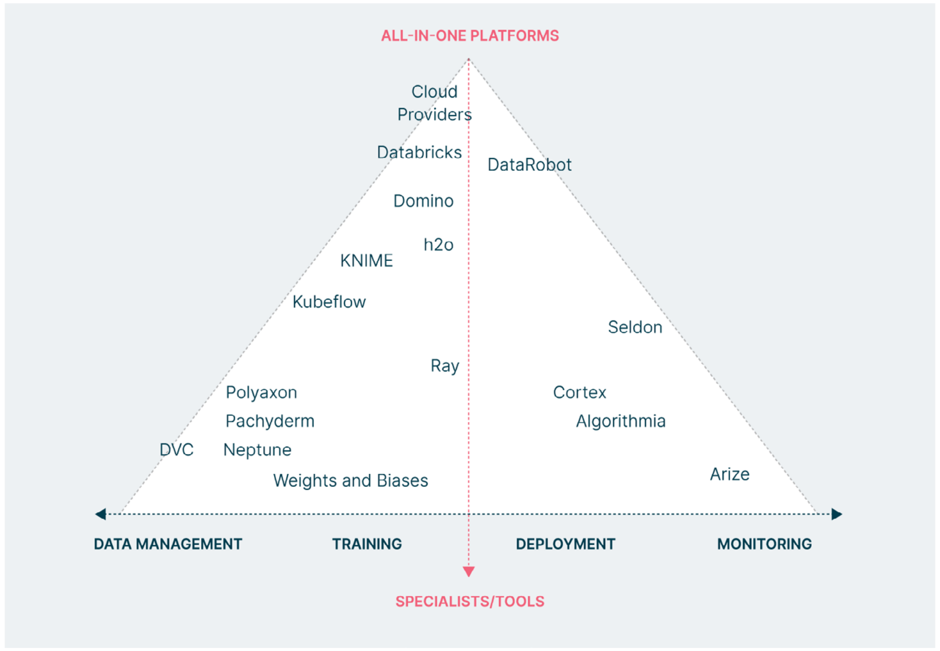


Figure 7: MLOps software as a pyramid with tools targeted at lifecycle stages and all-in-one platforms targeting all stages (but mostly with a certain leaning)

The products on the bottom-right are focused on deployment and monitoring; those on the bottom-left focus on training and tracking. Those at the very top do the whole spectrum. Those in the middle-top do most or all of the spectrum with a leaning one way or another.

Another way to visualize this is in terms of poles of specialization:

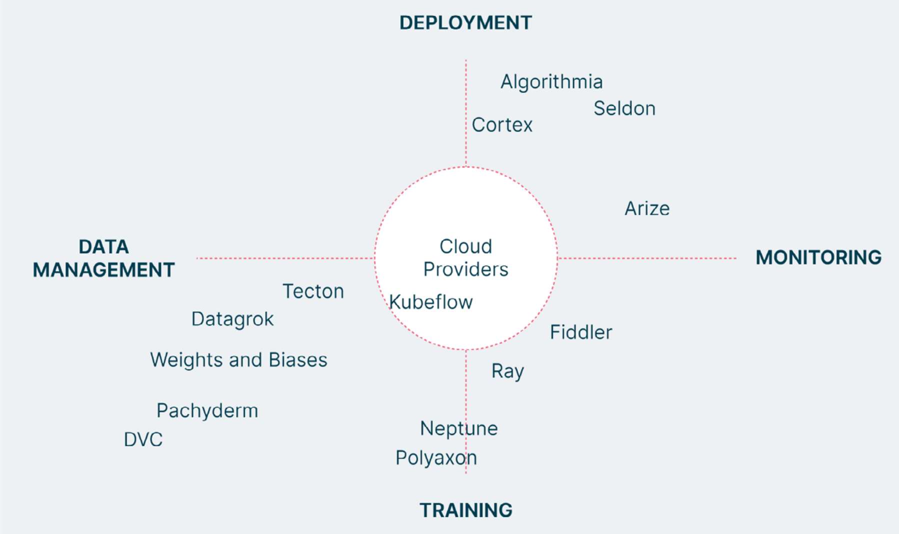
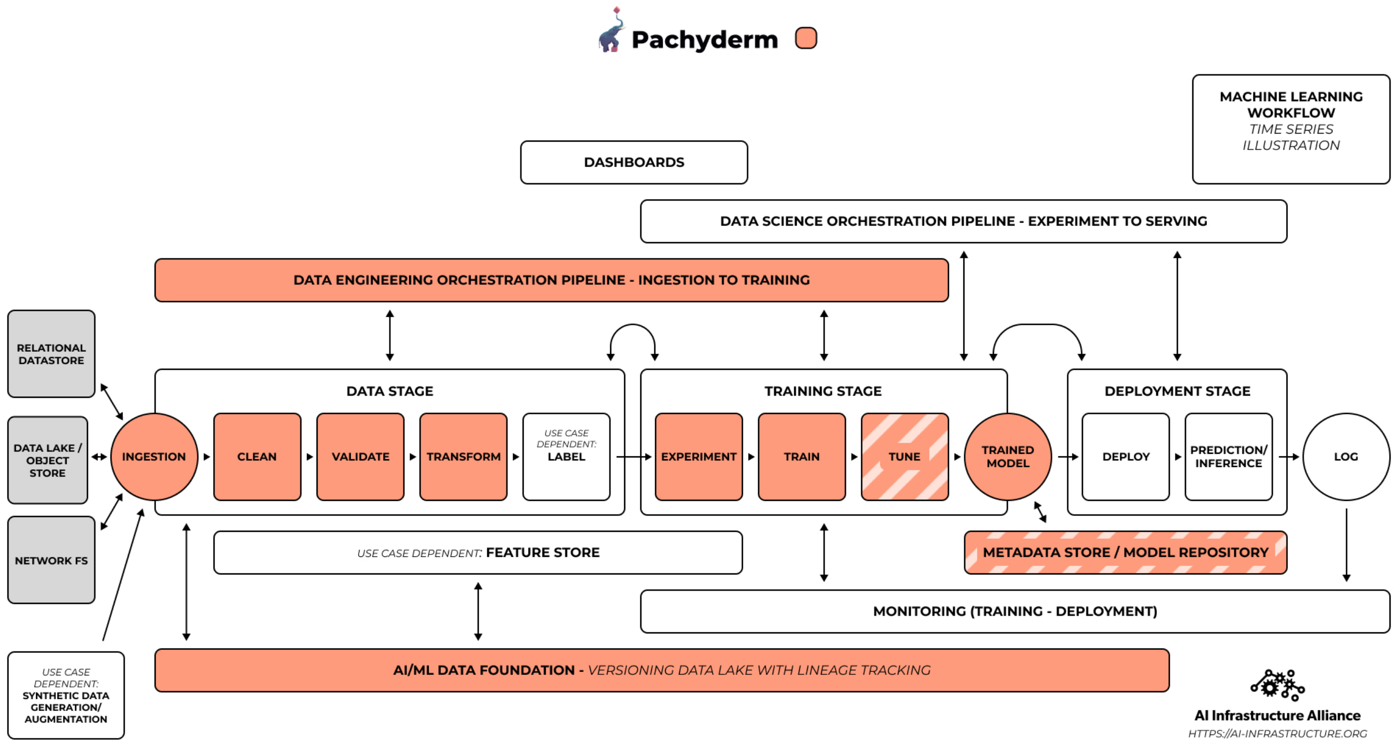


Figure 8: MLOps software with poles of specialization in lifecycle stages and all-in-one platforms gravitating towards poles (or middle if no particular pole)

In this way of thinking, the most balanced all-in-one platforms are in the middle if they cover different aspects equally. Specialized products are closest to their pole of specialization, though many specialize in more than one thing.

Here, for example, is a diagram from the AI Infrastructure Alliance illustrating which parts of an MLOps architecture are covered by Pachyderm (the parts coloured orange):



Pachyderm
MACHINE LEARNING WORKFLOW TIME SERIES ILLUSTRATION
DASHBOARDS
DATA SCIENCE ORCHESTRATION PIPELINE - EXPERIMENT TO SERVING
DATA ENGINEERING ORCHESTRATION PIPELINE - INGESTION TO TRAINING
RELATIONAL DATASTORE
DATA LAKE / OBJECT STORE
NETWORK FS
DATA STAGE
INGESTION
CLEAN
VALIDATE
TRANSFORM
USE CASE DEPENDENT: LABEL
TRAINING STAGE
EXPERIMENT
TRAIN
TUNE
TRAINED MODEL
DEPLOYMENT STAGE
DEPLOY
PREDICTION/ INFERENCE
LOG
USE CASE DEPENDENT: FEATURE STORE
METADATA STORE / MODEL REPOSITORY
MONITORING (TRAINING - DEPLOYMENT)
AI/ML DATA FOUNDATION - VERSIONING DATA LAKE WITH LINEAGE TRACKING
USE CASE DEPENDENT: SYNTHETIC DATA GENERATION/ AUGMENTATION
AI Infrastructure Alliance
HTTPS://AI-INFRASTRUCTURE.ORG

Figure 9: AI Infrastructure Alliance diagram showing which parts of an ML architecture that Pachyderm covers

Even the wide-ranging platforms have areas that they are stronger on and areas which are not their focus. We need to understand what the platforms are trying to achieve in order to compare them fairly or assess which is best for our situation.

Installation and integration models

MLOps platforms typically have a strategy for how they will be installed and what integrations they offer. Some key questions that can be used to categorize MLOps platforms are:

- Is the platform hosted as a PaaS or are there a variety of installation options?
- Are there integrations to popular tooling (e.g. source code management)?

These questions have a direct impact on the suitability of different platforms for different organizations. Imagine that an organization is evaluating the popular open source options of Kubeflow and MLflow. If the organization is using Kubernetes already, then Kubeflow being targeted at Kubernetes might be appealing, as might Kubeflow's distributions for different cloud providers. If the organization is not using Kubernetes, MLflow's more neutral approach might be more appealing.

Other organizations may not want to install and manage a platform themselves. For example, they may not want to deal with the low-level resource management or the need to handle updates. In such cases, a hosted PaaS might be more appealing.

For organizations that are committed to existing decisions about their technology stack, integrations can be a key concern. For example, it can be important that a platform integrates with their identity management system or continuous integration tools. Some platforms put more emphasis on integrations than others.

Some platforms have a 'one-stop-shop' strategy. They try to meet all MLOps needs in a single place. This can fit well with a PaaS model as organizations who want everything from one platform often don't want to operate the infrastructure for it. This model tends to be lighter on integration options.

Approaches and target personas — trade-offs of automation

Some platforms have particular approaches that they emphasize. This is most pronounced with AutoML. Here we can find the model building process structured around wizards and models created through code may be a secondary option or may not be supported.

Note that there is, once again, a spectrum here. Some platforms aim to reduce the level of knowledge needed to build ML models, targeting a citizen data scientist. Others use automation to reduce data scientists' workloads without taking away control over coding all the steps. Databricks, for example, has introduced an AutoML feature which generates notebooks for the models that the AutoML trials. This way, data scientists can dive in and adjust the code used to build the model.

Tensions around automation and flexibility for AutoML have been much discussed. Less discussed are similar tensions around automation in the deployment and monitoring phases. Some platforms emphasize deployments happening with a push of a button rather than deployment to require writing wrapping code[5] or an approval flow in source control.[6] Out-of-the-box monitoring may not be sufficient for all models and teams may seek to write custom alerts or dashboards or even the option to integrate with a separate monitoring tool. We know many data scientists value being able to delve into the code at the model-building stage, similarly, machine learning engineers can need points of customization and integration at the deployment and monitoring stage.

[5] A common use case for wrapping code is when the form of the data in the request to the model doesn't match the form that the model was trained on. Code is then needed to transform the request before it is passed into the model for prediction.

[6] Note that platforms have REST APIs but not all APIs are easy to use from CI.

We have a tension in MLOps between automation and flexibility. Wizards can simplify and reduce the knowledge and effort needed to perform a task. Wizards can be made more flexible with configuration options but don't offer the same flexibility as writing code. Most platforms attempt to support multiple paths and provide both automation and flexibility to different personas and skill levels. However, many platforms also tend to emphasize a particular approach. We can picture this in terms of poles of specialization:

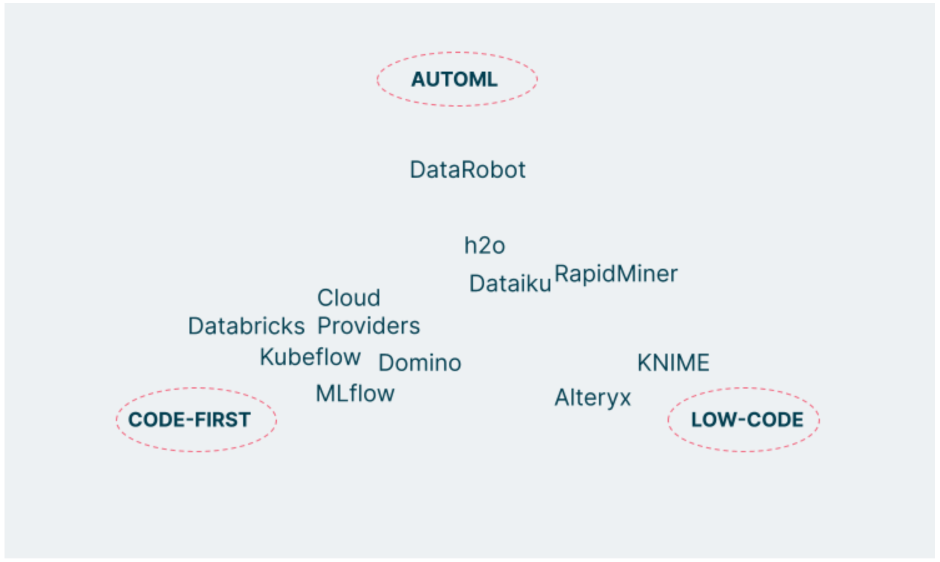


Figure 10: MLOps software with poles of specialization in approach with vendors gravitating towards poles (or middle if no particular pole)

In a code-first approach, data scientists can create models using whatever code-based tools they choose. An AutoML approach uses wizards to configure what search the platform will perform to find the best model to fit the supplied data. A low-code approach to model building provides a visual drag-and-drop environment to create the model training workflow from configurable steps.[7] Most vendors offer a mixture of these approaches, with many putting most emphasis on one of the three.

[7] A low-code environment may aim to reduce the skills needed for the tasks but this is not always the emphasis as there can be other attractions to a visual environment.

Choosing the right MLOps platform for you - avoiding the pitfalls

Choosing the right MLOps platform for you - avoiding the pitfalls

Not every company getting value from machine learning is using an MLOps platform. It is certainly possible to get models into production without a platform — an O'Reilly survey found upto 46% may have been deployed without using any available MLOps deployment tool. In some cases selecting and introducing a platform for a single project may be an unnecessary overhead. Platforms tend to be of most value where there are multiple projects, in that way skills and knowledge can then be shared more easily.

It can be difficult to decide on the best buying criteria for a platform and it can be tempting to look for 'completeness' of features. We believe it is risky to allow breadth of features to dominate considerations because:

- All platforms are adding features over time
- Range of features can come at a cost of inflexibility
- Many use cases don't require a wide range of features from a platform
- Where required features are missing, it is often sufficient to stitch together tools
- The variation of use cases makes 'complete' impossible to define in a neutral and detailed way

You might worry that your company has such a range of cases that you'll need everything. We find it best to perform a study, to put the cases down in writing, even in the biggest of companies. If the list is large, then it should be prioritized. That said, the breadth of features in a platform is not irrelevant either. If no single platform meets all your needs then you need to stitch together different tools. Stitching tools together is fine as long as it goes smoothly. But tools that aren't designed to work together don't always work well together.

Some companies have widely differing use cases which makes it difficult for a single platform to accommodate them all. Imagine that you have some combination of:

- Financial use cases with high governance and transparency requirements
- Long-running batch text processing use cases
- Models used in a high-traffic website with the need to A/B test versions in production
- Reinforcement learning use cases
- Low-latency prediction use cases requiring GPUs for inference
- AutoML use cases

You might go for a general-purpose MLOps platform to cover much of this, however you might need a plugin to a specific serving/monitoring solution for the high traffic and financial cases; the batch parts might be a better fit for Spark or Airflow; and if the platform doesn't have adequate AutoML then you could get that from a different platform.

You want to avoid a ‘zoo’ of different approaches being used by different teams. A zoo can easily emerge from different teams picking tech independently:

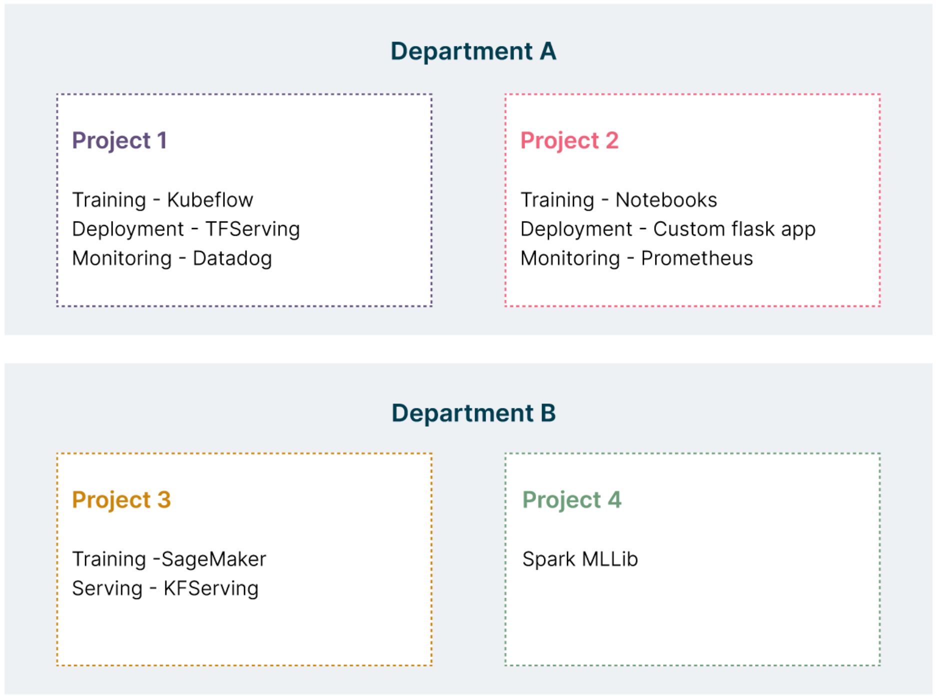


Figure 11: Risk of ‘zoo’ of MLOps tools

A platform should standardize and facilitate porting skills between teams. But one also has to be careful about restricting what teams can do. We recommend identifying a representative range of use cases and running PoCs on a small set of platforms (unless it’s already very clear that a case is or is not fully supported). This can help identify if a platform is not flexible enough for your needs.

Advice for structuring an MLOps platform evaluation

Advice for structuring an MLOps platform evaluation

An MLOps platform initiative typically has an owner — somebody with overall responsibility for guiding the decision-making. Depending upon the size of the organization, the owner may be supported by a dedicated team. The stakeholders may be represented on a steering committee to oversee the evaluation.

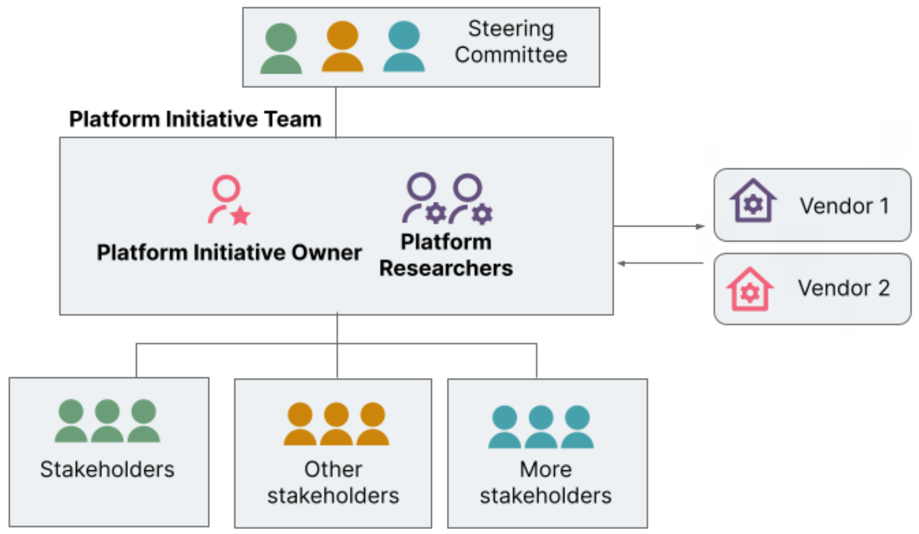


Figure 12: Composition of Actors in an MLOps Platform Evaluation

Artefacts produced in an evaluation include:

- A roadmap for narrowing down options and target decision dates
- Interviews with key stakeholders
- Important user journeys
- Key use cases to be handled
- An initial comparison that narrows the field from a long-list of options. Options might include an in-house build based on open source tools. There might also be incumbent setup to be compared against
- A write-up of any PoC performed on short-listed options
- Detailed evaluation of short-listed options to make a final selection

The final decision may be made by the owner or by a committee but the owner will be responsible for the material used to make the decision. There may also be an implementation roadmap (covering integration with other systems). Some integration work may be needed during evaluation for PoCs or to prove integration feasibility.[8]

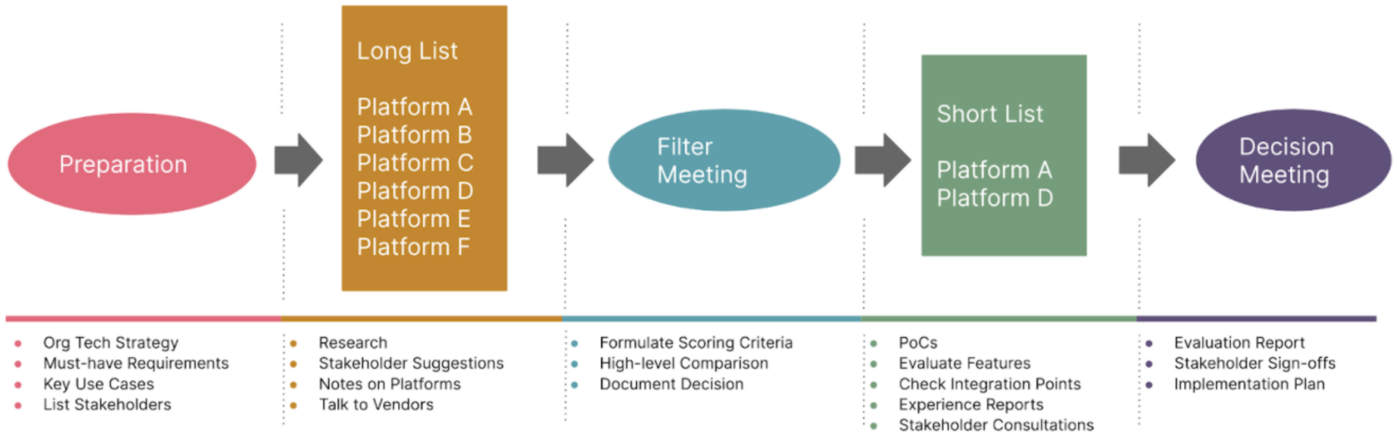


Figure 13: Flow of an MLOps platform evaluation with artefacts and activities

We suggest structuring an evaluation around these phases:

- Selecting the contenders for the long-list
- Narrowing the long-list down to a short-list
- Performing a detailed evaluation to make a selection

When selecting the contenders for the long-list, we recommend considering these key factors:

- Your cloud/technology strategy
- Alignment to other tools in the organization's tech stack
- Knowledge and skills in the organization[9]
- Key use cases and/or user journeys
- Support arrangements

[8] We've covered points of integration to data and monitoring systems in 'Purpose and Variation of MLOps Platforms'. Other key points of integration include identity management, authorization, source control and continuous integration systems.

[9] In a larger organization a survey might be necessary to assess existing skills/ knowledge.

For the first selection you will go on high-level impressions. This requires a high-level understanding of what platforms are out there and what they do. You will want an overview-level understanding of the platforms and how each is differentiated at a strategic level.

The next phase of comparison is about going deeper. You may also add extra criteria such as pricing and ease of implementation/migration. If there are particular use cases for which it's unclear how you would use one of the short-listed platforms, then PoCs can be run to flush out any issues. PoCs can also be run and documented as experience reports to inform usability ratings.

You could use a high-level table like this:

	Key Criteria						Nice to haves	
Tool	Fit with cloud/infra strategy	Support for key NLP case A	Knowledge in team	Knowledge in team CI/CD Integration	Pricing	Gaps?	Support in timezone X	Integration to Analytics Platform
Platform A	5	2	1	3	4	Image support limited	Yes	3
Platform B	3	5	4	4	2	Batch support limited. Use Spark for batch?	No	2
Platform C	2	2	2	3	2	Data prep limited	Yes	2

Figure 14: Example comparison matrix for an MLOps platform decision

The above is just for illustration — your criteria will vary. This format is just one high-level lens of comparison.[10]

A high-level table like the one on the left is important as a summary but it will likely not provide all of the detail you need. You will want to reference any write-ups or experience reports from PoCs. You will need to get into detailed feature comparisons in order to assess whether your key use cases are met. You might do feature comparisons on separate tabs of a spreadsheet or reference them within an evaluation report document.

We suggest creating your own feature comparisons rather than copying from a website. For this purpose **we provide open source material that can be used to compose feature comparisons.** We explain how to use this material in more detail in the next section.

[10] We don't cover all aspects of an evaluation or scoring matrix. We're not going into the various ways software products can be compared against each other. We don't debate the relative merits of scores vs Harvey Balls.

How to use open source material to bootstrap your evaluation

How to use open source material to bootstrap your evaluation

We have launched open source material to bootstrap your evaluation in both the early selection phase and the detailed evaluation phase:

https://github.com/thoughtworks/mlops-platforms

For the early selection phase, we explain the product strategy and architecture of a number of platforms. We offer breakdowns of popular platforms and explain their key concepts. This is key to understanding whether the product might be a general fit for your organization (high-level technology choices and ways of working).

We also offer a detailed feature comparison matrix structured around high-level categories. Unlike other comparison matrices, ours has an open format with rows as categories rather than features. We put references to product features (and references to the product documentation) within those categories. This gives vendors the scope to do things their own way. It also enables readers to build tailored comparisons from our material.

In the detailed evaluation phase, you need to decide whether a platform can work for your use cases. In the course of reviewing platform features and conducting PoCs, you may discover unexpected nuances to your use cases or even whole use cases that were not anticipated. We want to facilitate this discovery process by helping you to explore detailed product features. This gives you the opportunity to find out about features that you hadn't previously considered.

Here is an shortened example row from our matrix as of October 2021 (shortened to just three platforms for readability):

	AWS SageMaker	Azure Machine Learning	Google AI Platform (Vertex)
Model Governance - model registry/ catalogue	Model registry with versions, groups and associations to training metadata. Can enable cross-account deployment.	Per-workspace registry and shareable across workspaces.	Not an explicit concept in docs but there is a models page in a vertex project

Here we could have chosen to treat 'model registry' as its own row with a tickbox. But that would lose the nuance that the capabilities of the registries are different and that Vertex has relevant capabilities that are not called a 'registry'. It would also lose the link to the documentation for readers to inspect the feature for themselves.

We also do not offer scores to grade features against each other. You know your specific criteria, so you can take our material and use it to construct your own scores or tick boxes as applicable (see the example table with scores in the previous section).

You can find many ways of doing feature comparisons online that claim to determine which is the 'best' option. We caution against this as there are many dimensions by which to assess features and all of them are biased. This is easily missed as it's necessary to see the dimensions of a feature area before the bias is visible.[11] You should create your own comparisons because those tables will be biased towards your needs.

We are not trying to find 'the best MLOps platform'. **We want to help organizations buy/build/assemble an MLOps platform that can best serve their particular needs.**

[11] Just for experiment/training management, there are dimensions like range of languages supported, usability of GUI, CLI support, API security models, roles and permissions, error handling model, retry model, scalability, etc. A row for each of these is too much detail to take in but simplification to one or two tickboxes leads to bias.

Trends and Analysis

Trends and Analysis

The MLOps scene is a very dynamic one. We can see this from the hundreds of tools becoming available but also from the range of significant new features being added by the big players. These new tools and features offer users new opportunities. They also present a hazard to those looking to standardise ways of working across an organization. We aim to limit future rework but as industry practises change some rework may be unavoidable.

The clearest recent trends (at the time of writing) evident in the big platforms are:

- Deployment and monitoring. Azure launched serving features in 2021 and Databricks in 2020. Google launched key monitoring features linked to serving in 2021. DataRobot acquired deployment platform Algorithmia in July 2021.
- Feature stores. Databricks and Google both launched feature stores in 2021. AWS announced theirs at the end of 2020.
- Tooling focused on data wrangling. AWS Data Wrangler was announced at the end of 2020 and Microsoft's Synapse integration is in preview. DataRobot acquired Paxata for data prep at the end of 2019. Several of the vendors featured in comparisons are offering ways to make data wrangling visual and interactive.
- AutoML offerings are getting more diverse, branching out into a range of assisted ML features for experienced data scientists as well as approaches aimed at empowering citizen data scientists.

These are trends to watch for in the future as vendors will likely deepen these new offerings and other platforms will likely follow.

The platform vs tools divide

One characteristic of the MLOps field is a platforms versus tools divide. This leads to organizations having to choose from the following options:

- Buy a platform and patch with tools/in-house for any parts that don't fit
- Assemble a platform from the range of MLOps tools
- Pause on introducing a platform and handle cases as they come for now

Each of these options has risk associated with it. This guide focuses on platforms but we do not mean to recommend the off-the-shelf platform option for everyone. An O'Reilly survey found up to 46% may have been deployed without using any available MLOps deployment tool. Platforms are significant but don't appear to be dominant. The trend from tools and startups appears to be that the whole field is growing and that particular parts are growing a little faster than others (diagram from Chip Huyen):

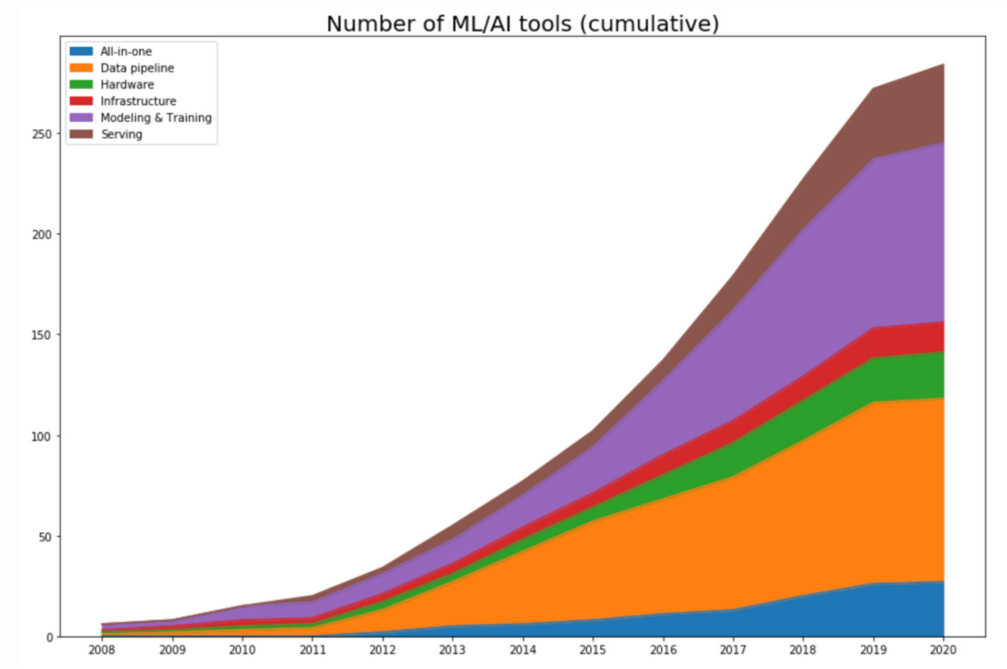


Figure 15: MLOps tools by year of launch and color-coded by category.

Diagram and data from Chip Huyen

Riding the waves of change

One might ask if there's a way to protect against change by simply following a big trend and picking a 'winner'. But there is no clear trend towards winners right now.

It's tempting to try to speculate about the future of MLOps by analogy with what has happened in related fields, especially fields under DevOps. Google certainly seems to be going for a similar play with Kubeflow to its play with Kubernetes. The aim seems to be for the open source to gain widespread adoption while Google also puts resource into offering a managed version of Kubeflow. However, there are also differences. Vertex encompasses more features than Kubeflow and has some features that overlap with Kubeflow but do not come from Kubeflow. Further, it is not clear that the dynamics of the MLOps space are like the dynamics of the container orchestration space.

One might ask whether end-to-end platforms will take much of the MLOps market when the DevOps space has so many different tools and vendors. There are certainly resemblances between some DevOps tools and some MLOps tools. For example, a model registry resembles artifact stores such as artifactory and MLOps training systems resemble CI systems (though there is difference as well as resemblance). Does that indicate that the market for training platforms will resemble that for CI systems? It's not clear that the range and market share of CI platforms we have now is due to the dynamics of CI. It could owe much to accident.

A better comparison for MLOps tools right now is perhaps the infrastructure as code space. Here we see a handful of tools (e.g. Terraform, Ansible, CloudFormation) that both compete with each other for some purposes and can be used together in others. This puts a burden on users to learn the range of tools but the burden is reduced if the tools work in similar ways.

We might therefore expect tools to eventually have more fundamental areas of similarity. It should over time become clearer in what areas to expect tools to be similar to one another and in what areas they might differentiate.

A further caution for speculating about the future of the MLOps field is the role of AutoML. When pitched at citizen data scientists, this could be regarded as its own market with a dynamic that has similarities to the low-code/no-code platforms market. But some AutoML is also pitched as assisted automation for experienced data scientists. This will continue to influence the shape of MLOps. This is one of a number of questions that need to be played out through practice. [12]

We caution against seeing the market too much in terms of a battle between specialist tools and all-in-one platforms. **MLOps needs are varied.** Not everyone needs a whole platform. Some specialist tools complement a platform. If a specialist tool really takes off, it's likely big vendors will look to incorporate or interoperate with it. We don't currently see signs of either platforms or tools becoming a default form.

There is a race to get value from AI and MLOps is a key enabler of that. Organizations must make decisions that work best for them right now and avoid the temptation of the crystal ball. Change is unavoidable but the impact of change will be lessened by others going through the same change together.

[12] There are many open questions to be played out in the coming years. Will feature stores continue to grow into the mainstream? Will model registries become standard even though they don't fit for reinforcement learning? Will a standard for model packaging emerge that accommodates the range of frameworks and allows for custom transformation functions? Is achieving data lineage always worth the overhead in both effort and storage cost? Are visual data wrangling tools positioned to take over from writing wrangling code? Are there ways to incorporate AutoML that still give data scientists the control they desire? When is explainability worth the tradeoffs? Which roles should have permission to deploy? How much knowledge of the data should be expected for ML monitoring? When is it worth capturing all data going through the system?

About the authors

Lead author: **Ryan Dawson**

Principal Consultant, Data and AI

A technologist passionate about data, Ryan works with clients on large-scale data and AI initiatives. Ryan helps organizations get more value from data. As well as strategies to productionize machine learning, this also includes organizing the way data is captured and shared, selecting the right data technologies and optimal team structures. He has over 15 years of experience and is author of many widely-read articles about MLOps, software design and delivery.

ryan.dawson@thoughtworks.com
@ryandawsongb, linkedin

Lead reviewer: **Danilo Sato**

Head of Data & AI Services for ThoughtWorks UK

Throughout his 20-year career, Danilo has combined his experience leading accounts and teams with a breadth of technical expertise across architecture and engineering, software, data, infrastructure, and machine learning. Danilo is the author of DevOps in Practice: Reliable and Automated Software Delivery, a member of Thoughtworks Technology Advisory Board and Office of the CTO, and an experienced international conference speaker.

dsato@thoughtworks.com
@dtsato, linkedin

Assisting contributor: Lucy Fang, Consultant
Design assistance: Lasse Tammilehto, Principal Experience Designer

Reviewers: Rebecca Parsons, Ken Mugrage, Sathyan S Sethu Madhavan, Max Pagels, Krishna Meduri, Ricardo Savii, Meissanne Chami, Jepson Du

About Thoughtworks

Thoughtworks is a leading global technology consultancy that integrates strategy, design and software engineering to enable enterprises and technology disruptors across the globe to thrive as modern digital businesses. Thoughtworks has helped numerous clients accelerate their AI journeys by practising CD4ML, an end-to-end approach to MLOps. CD4ML was born in 2016, when Thoughtworks built a pricing recommendation engine with CD4ML for AutoScout24, the largest online car marketplace in Europe. Since that, we have helped clients implement organization-wide AI initiatives using commercial platforms and in-house platforms. We've also built many models and taken them to production without an organization-wide platform. We like to understand your specific challenges and help to solve them in a way that fits your organization.