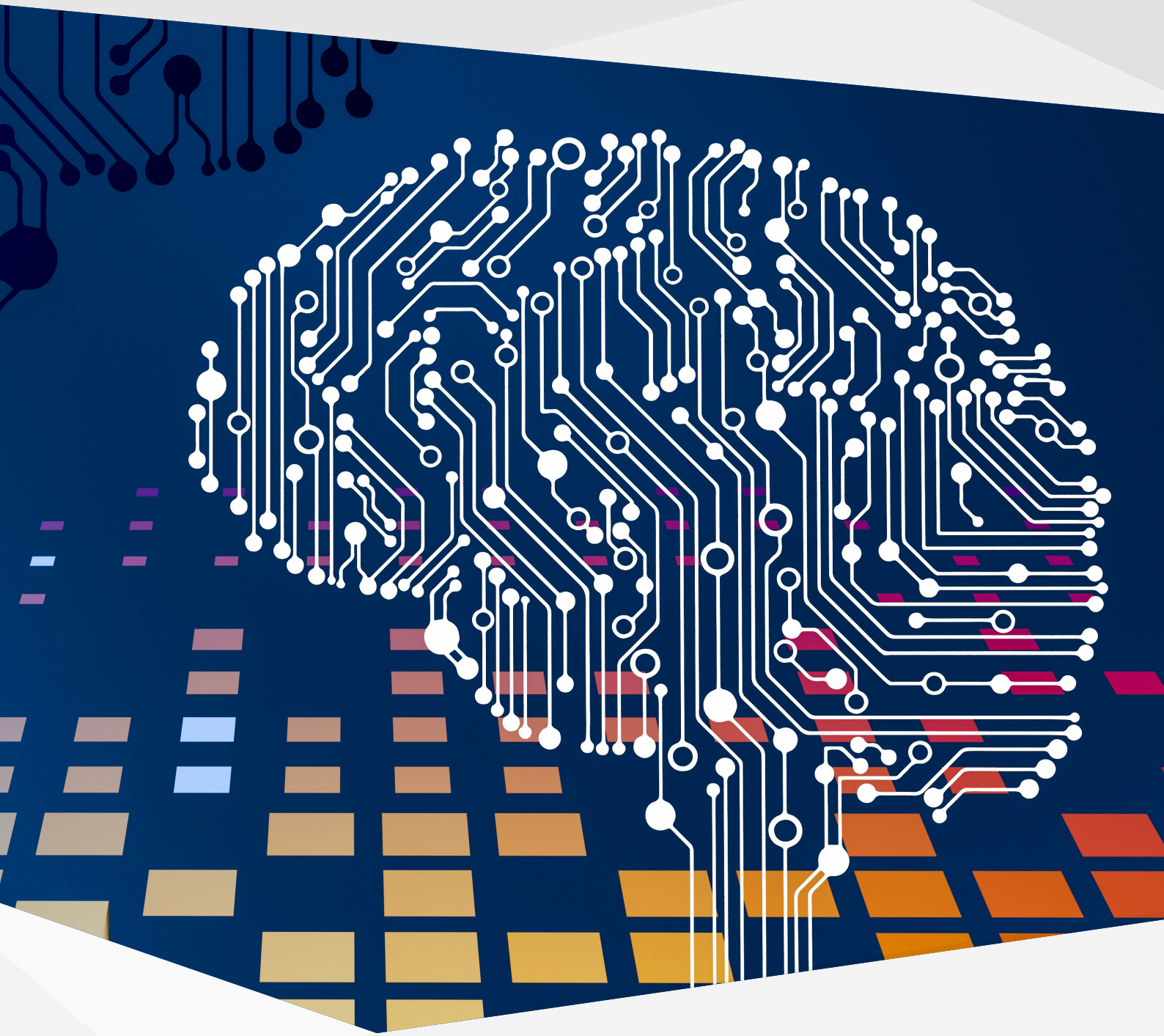
Canonical

Lightweight machine learning with MLflow



Executive summary

The latest advancements in AI – such as the release of Generative AI Foundry on Microsoft Azure and launch of open source large language models like Llama 2 and Mistral – not only made the tour of the newspapers, they also challenged organisations to rethink their AI strategies. For many businesses, AI is becoming the core of their journey towards digital transformation, as well as the central driver accelerating innovation across the board. From DNA sequencing in life sciences to drill summarisation in oil & gas, enterprises are now using machine learning to tackle some of their most pressing and long-standing problems.

Whereas AI's potential is evident to business leaders, data science professionals and laymen alike, often organisations get stuck due to project complexity or the fast-changing nature of the AI landscape. As a result, enterprises can struggle to get started and end up postponing even preliminary work. The best way to mitigate this risk is by choosing the right tooling.

Any successful AI starts with a problem that needs to be solved, as well as enough data to develop machine learning models. Once you have these, the next step is to start looking for the most suitable tools. There are a multitude of platforms nowadays on the market to help enterprises, whether they are open source or proprietary. However, many of them require considerable resources or advanced skills in machine learning, and so tooling can become a blocker for organisations that are in an exploratory phase and need an accessible, lightweight machine learning platform.

MLflow is the ideal solution for these use cases. It is an open source machine learning platform uniquely suited to lightweight ML thanks to its flexibility and ease-of-use. In this whitepaper, we explore the central concepts of MLflow, as well as its primary components. We examine the main reasons to choose MLflow and key factors to look at when comparing MLflow with other MLOps platforms.

Contents

Machine learning platforms

The machine learning industry has evolved tremendously in the last couple of years. This has led to the development of a large number of platforms that cover different needs and address users with different experiences. Many of them are open source.

Choosing the right platform for a machine learning project is still a challenge for most professionals. The landscape is quickly changing and new capabilities are introduced with dizzying speed. It's easy to feel overwhelmed by the numerous options on the market, and it can be difficult to determine which platforms are reliable. Some of the key factors to consider when selecting a machine learning platform are:

- **Project scale**: Whereas tools such as Kubeflow are ideal for running AI at scale, JupyterNotebooks or MLflow are more suitable for local, more exploratory projects.

- **Stage of the machine learning lifecycle**: When choosing a platform, take into account what areas of the ML lifecycle you would like to cover. For example, Seldon is great for model serving or model monitoring, but it will bring little value to pipeline automation.

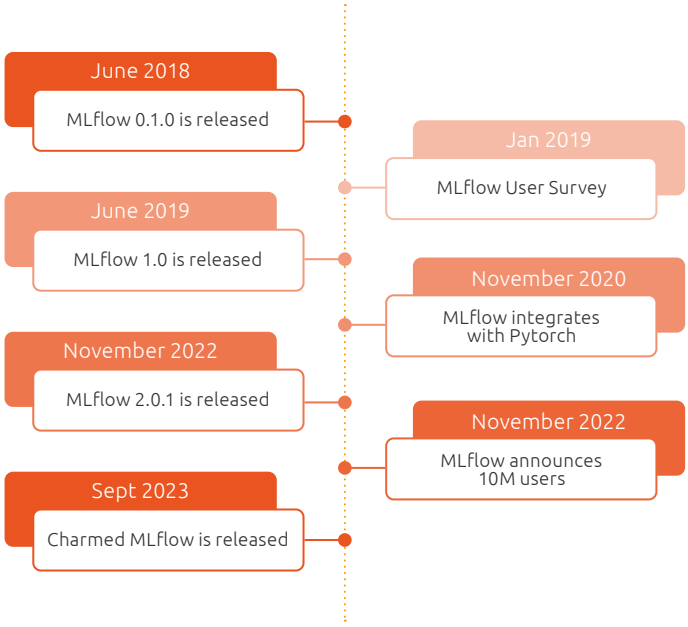
- **Cloud environment**: Depending on where you plan to run your project, you will need to look at the different solutions. For example, MLflow runs smoothly on Ubuntu, on the public cloud or on-prem, but solutions such as SageMaker are compatible only with the public cloud. Furthermore, if your project is part of a wider initiative, bear in mind scenarios such as hybrid cloud or multi-cloud.

- **Open source or proprietary tools**: While proprietary tools can be useful in some situations, open source solutions are ideal for getting started quickly, while still giving you the option to scale into production with enterprise support.

- **Integrations needed**: This applies especially to industries or projects that run on data very specific to their activity. The machine learning platform needs to be able to seamlessly integrate with the highly-specialised frameworks and libraries involved in these use cases.

What is MLFlow?

History of MLflow



MLflow is an open source machine learning platform for the machine learning (ML) lifecycle. It was started in 2018 and it has kept growing in popularity ever since. In November 2022, it reached 10 million users and nowadays it is used by important players from the AI world such as DataBricks, Meta, HuggingFaces and Accenture.

MLflow appeared on the market as a solution for experiment tracking but it evolved over time into a handy, easy-to-use solution that has many capabilities that help with code reproducibility and project packaging. Nowadays, MLflow is an expandable, open-source platform for managing workflows and artefacts across the machine learning lifecycle. It has built-in integrations with many popular ML libraries but can be used with any library, algorithm, or deployment tool. It is designed to be extensible, so you can write plugins to support new workflows, libraries, and tools.

Why use MLflow?

How to use MLflow

As a Data scientist - you track your experiments.

As a MLOps engineer - you compare the performance of different models.

As a ML engineer - you get started locally and structure your project.

As a ML team - you integrate it with other tools depending on the use case.

As a AI enthusiast - you contribute to the upstream project to make MLflow better.

MLflow, similarly to any other tool, was built to solve a problem. Experiment tracking was one of the biggest challenges facing data scientists and ML engineers, and it was addressed by MLflow. The open source platform helps users reduce time spent on analysing results and more quickly track the best performant experiments. However, there are many other reasons to use MLflow:

- **Results reproducibility**: Even when starting with a simple plan, the complexity of ML projects tends to escalate quickly, resulting in an overwhelming quantity of experiments. Manual or non-automated tracking leads to a high chance of missing out on finer details. ML pipelines are fragile, and even a single missing element can throw off the results. MLflow enables automated tracking to significantly mitigate this risk and ensure that experiment results are reproducible.

- **Works anywhere**: Many machine learning tools require a lot of resources to run. This is a huge blocker, especially for early adopters or projects that are in the experimentation phase. MLflow breaks this trend, as it can work anywhere. Whether you are using an average-priced laptop or complex architecture designed to run AI at scale, the open source tool is versatile enough to adapt.

- **Environment agnostic**: MLflow runs on any infrastructure – including private clouds, public clouds or more complex hybrid or multi cloud scenarios. It also works on top of Kubernetes platforms. For example, Charmed MLflow is built as a cloud-native application that works on any CNCF comformant Kubernetes distribution.

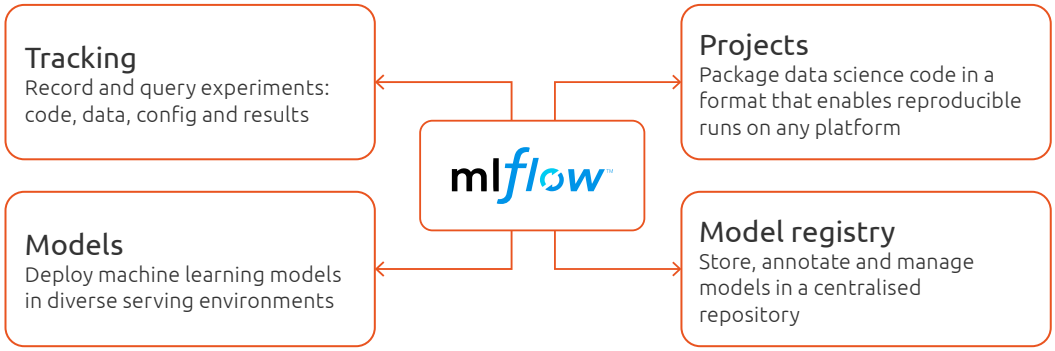
- **Ease-of-use**: In contrast to the majority of ML tools that are difficult to use and designed for advanced users, MLflow offers ease-of-use as a key differentiator. MLflow includes an intuitive user interface (UI) so users do not need to rely exclusively on the command line interface (CLI), making it especially accessible to beginners. MLflow's ease-of-use is a key factor behind its widespread adoption.

- **Tools integrations**: The difficulty of use is not the only problem that other ML tools face. The machine learning landscape is still very scattered, with software that mostly works independently and forces users to make constant choices and tradeoffs. Conversely, MLflow is a platform built to smoothly integrate with other tools. It works seamlessly alongside frameworks such as PyTorch or Tensorflow, big data tools such as Apache Spark, MLOps platforms such as Kubeflow or libraries such as HuggingFaces.

- **Enterprise support or managed services**: Open source tools are a great asset to effectively drive an organisation's AI journey. However, organisations often want to rely on support from experts or offload the burden of maintaining the infrastructure needed to develop and deploy ML models. Charmed MLflow is just one of the distributions that, while fully open source, offers enterprise support or managed services. Additionally, it enables timely bug fixes, as well as security patching.

What's inside MLflow?

Modern MLflow has four primary functions: experiment tracking, model registry, model management and code reproducibility. It is especially suitable for beginners since it requires few resources to be deployed and it can run everywhere. It is often integrated with other tools such as Apache Spark, Jupyter Notebook or Kubeflow to be used at different scales and for different tasks.



MLflow Tracking

Tracking allows you to monitor experiments to record and compare parameters and results. It is used to track different pipeline parameters such as metrics, hyperparameters, feature parameters, code versions, and other artefacts. You can use these logs to visualise or compare the results between experiments, users, or environments. The logs can be stored both on any local system and remote servers.

You can use this component to log various aspects of a run, including:

- **Source**: The name of the file that launches the run. Alternatively, if you are using an MLflow project, it can be the name of the project and entry point of the run.
- **Code version**: When using an MLflow Project, this would be the Git commit hash.
- **Parameters**: Any key-value input parameters you choose, as long as the values and the keys are both strings.
- **Artefacts**: Output files (in all formats).
- **Start and end time**: Record the start and end time of your run.
- **Metrics**: Key-value metrics containing numeric values. It is possible to update each metric throughout a run. It helps track how the loss function of the model is converging. Additionally, MLflow lets you visualise the full history of each metric.

Whereas at first sight, this information seems tedious to track, it comes in handy later in the machine learning lifecycle, when visualising the results of each run as well as when analysing the experiment as a whole. To more easily visualise, compare, and search runs, you can take advantage of the user interface. MLflow tracking capabilities also go a step further, enabling you to download the metadata and the artefacts for runs, which you can input for analysis in other tools.

MLflow Models

Models allow you to manage and deploy models from a variety of ML libraries to a variety of model serving and inference platforms. With MLflow Models, your ML model can be packaged into different 'flavors.' A 'flavor' is a format or structure such as a TensorFlow DAG or a Python function, and the descriptor file defines it. This ability to package 'flavors' enables the model to be used across a host of downstream tools and platforms such as on Docker or AWS SageMaker, and consequently makes the model lifecycle easier to process and manage.

Metadata that can be added to the models include:

- **Model signature**: Defines the schema of the inputs and outputs of your model. Signatures are stored as JSON in the MLmodel file, along with additional model metadata.
- **Model input example**: An artefact that is an instance of a model input, stored together with the model.

MLflow Projects

Projects allow you to package ML code in a reusable, reproducible form to share with other data scientists or transfer to production. It offers a convention for packaging or structuring your ML projects and reusable project codes. Fundamentally, a project is a directory along with a descriptor file that defines the structure and dependencies. Additionally, on using the MLflow API in the project, MLflow automatically remembers the parameters or project details.

The main properties of an MLflow project are:

- **Name**: It is advised to use human-readable names for your projects.
- **Environment**: Define the software environment used to execute the entry points of the project, including all library dependencies required by the project code.
- **Entry points**: Specify the commands you want to run inside the project, as well as information about the parameters. A project typically contains at least one entry point that users can call.

You can run a project using the Git URI. Alternatively, you can use the MLflow run command line to run the project from a local directory.

MLflow Model registry

Model Registry enables you to centralise a model store for managing models' full lifecycle stage transitions: from staging to production, with capabilities for versioning and annotating. Databricks provides a managed version of the Model Registry in Unity Catalog. MLflow Registry acts as a core and enables APIs, UI, and centralised model storage. It aims to govern the end-to-end ML pipeline through tracking model lineage and versioning capabilities.

There are a couple of key features to bear in mind when using the model registry:

- **Registered model**: A model that has a unique name and metadata, contains model versions and transitional stages, and has a model lineage.
- **Model version**: Registered models can contain one or more model versions. When a new model is registered, it is considered version 1. Any new model using the same name is added as a subsequent version.
- **Model stage**: Each model version can have one stage assigned at any time. They need to be assigned according to the officially determined MLflow stages, such as staging, production, and archived. It is possible to transition a model version from one stage to another.
- **Annotations and descriptions**: You can add information that is relevant long-term. Annotations are done using markdowns. Descriptions can contain any information, including algorithm descriptions, methodology and dataset employed.

The two key MLflow concepts

MLflow is a very simple platform, designed to run lightweight machine learning. It is built around two important concepts: runs and experiments. They are the core of the solution and represent the key factors in proficiently using the platform.

A run is a collection of parameters, metrics, labels, and artefacts related to the training process of a machine learning model. They are the most basic concepts of MLflow. Any data scientists will have multiple runs as part of their activity, to identify the most performant model.

An experiment is a collection of runs. It lets you visualise, search for and compare runs, as well as download run artefacts and metadata for analysis in other tools. Similarly to runs, a data scientist will perform multiple experiments as part of their job.

Kubeflow vs MLflow

Kubeflow and MLflow are both leading platforms in the machine learning industry. While they were initially designed for different purposes, with some complementary capabilities, over time they have evolved overlapping features. This is why professionals have started comparing the two solutions. So, who wins in the battle between Kubeflow and MLflow?

What is Kubeflow?

Before comparing the two platforms, let's define Kubeflow. It is an open-source end-to-end MLOps platform started by Google a couple of years ago. It runs on any CNCF-compliant Kubernetes and enables professionals to develop and deploy machine learning models. Kubeflow is a suite of tools that automates machine learning workflows, in a portable, reproducible and scalable manner.

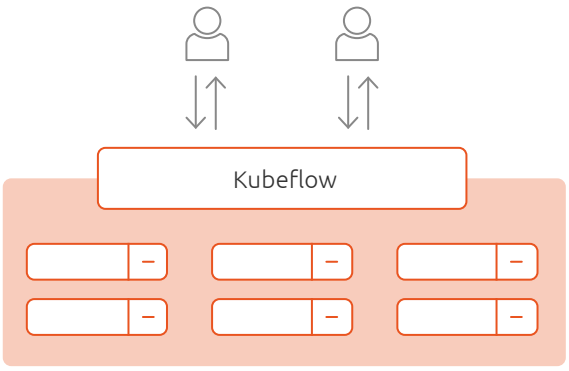
Kubeflow provides a platform for performing MLOps practices, with tooling to:

- Spin up a notebook
- Carry out data preparation
- Build pipelines to automate the entire ML process
- Perform AutoML and training on top of Kubernetes.
- Serve machine learning models using Kserve

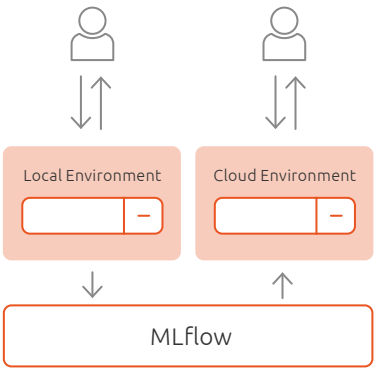
Similarities between Kubeflow and MLflow

Now going back to the comparison between Kubeflow and MLflow, there are some similarities between the two platforms. They are both open source solutions designed for the machine learning landscape. They both received massive support from industry leaders, and both are backed by large, passionate communities whose contributions are making a difference in the development of the project. Both tools integrate with other open source solutions from the industry to deliver a better experience for its users and address different stages of the machine learning lifecycle. The main purpose of Kubeflow and MLFlow is to create a collaborative environment for data scientists and machine learning engineers, empowering them to develop and deploy machine learning models in a scalable, portable and reproducible manner.

Differences between Kubeflow and MLflow



Kubeflow ArchitectureSimplified



MFflow ArchitectureSimplified

However, comparing Kubeflow and MLFlow is like comparing apples to oranges. From the very beginning, they were designed for different purposes. The projects evolved over time and now have overlapping features. But most importantly, they have different strengths. On one hand, Kubeflow is effective when it comes to machine learning workflow automation, using pipelines, as well as model development. On the other hand, MLFlow is great for experiment tracking and model registry. Also, from a user perspective, MLFlow requires fewer resources and is easier to deploy and use by beginners, whereas Kubeflow is a heavier solution, ideal for scaling up machine learning projects.

	Kubeflow	MLflow
Open source	Yes	Yes
Concepts	Container orchestration system, that enables training, model serving and inference	Python program where the training depends on the developer's choice
Target audience	Proficient users	Beginners
Use cases	Develop and deploy models at any scale	Experiment tracking locally on the developer's machine
Ease-of-use	There is a learning curve	Yes
Integrated Notebooks	Yes	No
Experiment tracking	No	Yes
Multi-tenancy	Yes	No
Pipelines	Yes	No (MLflow has pipelines, but they are in the early stage)
Model serving	External	Yes
Portability	Yes	Yes
Scalability	Yes	Yes
Tools integrations	Yes	Yes
Enterprise support	Yes	Yes
Managed services	Yes	Yes

Overall, Kubeflow and MLFlow should not be compared on a one-to-one basis. Kubeflow allows users to use Kubernetes for machine learning in a proper way and MLFlow is an agnostic platform that can be used with anything, from VSCode to JupyterLab, from SageMake to Kubeflow.

The best approach, if the layer underneath is Kubernetes, is to integrate Kubeflow and MLFlow and use them together. Charmed Kubeflow and Charmed MLFlow, for instance, are integrated, providing the best of both worlds. Using the two platforms together is easy and smooth, particularly when you follow our guide.

Conclusion

MLflow is an open source machine learning platform that runs anywhere and is easy-to-use. As a lightweight tool, it is perfect for getting started with AI. It benefits from the endorsement of an immense community and a high number of users, ensuring that there are plenty of tutorials and projects to play with.

The main capabilities of MLflow are built around experiment tracking, code and model packaging, and model registry. It integrates with a wide variety of frameworks, libraries and tools, including Kubeflow. For those who are interested in using MLflow as part of an enterprise-grade environment, Charmed MLflow is a platform that offers enterprise support and managed services.

As the machine learning world evolves, new capabilities will continue to be added to existing machine learning platforms. The industry is exploring the options to run AI using a lower compute, which also translates into a need to have more lightweight tools. Depending on the features that will be added as well as the integrations that will be enabled, MLflow is likely to see even wider adoption as the platform of choice for lightweight ML.

Sales Support

Contact us to build your MLOps solution, from lightweight environments to AI at scale.

Additional resources

Read more about Canonical MLOps

- Getting Started with AI
- MLOps toolkit explained
- Run AI at scale
- AI on Ubuntu

Canonical