KDnuggets

A Tour of End-to-End Machine Learning Platforms

<= Previous post
Next post =>

Tags: AirBnB, Data Science Platform, Google, Machine Learning, MLOps, Netflix, Pipeline, Uber, Workflow

An end-to-end machine learning platform needs a holistic approach. If you're interested in learning more about a few well-known ML platforms, you've come to the right place!

---

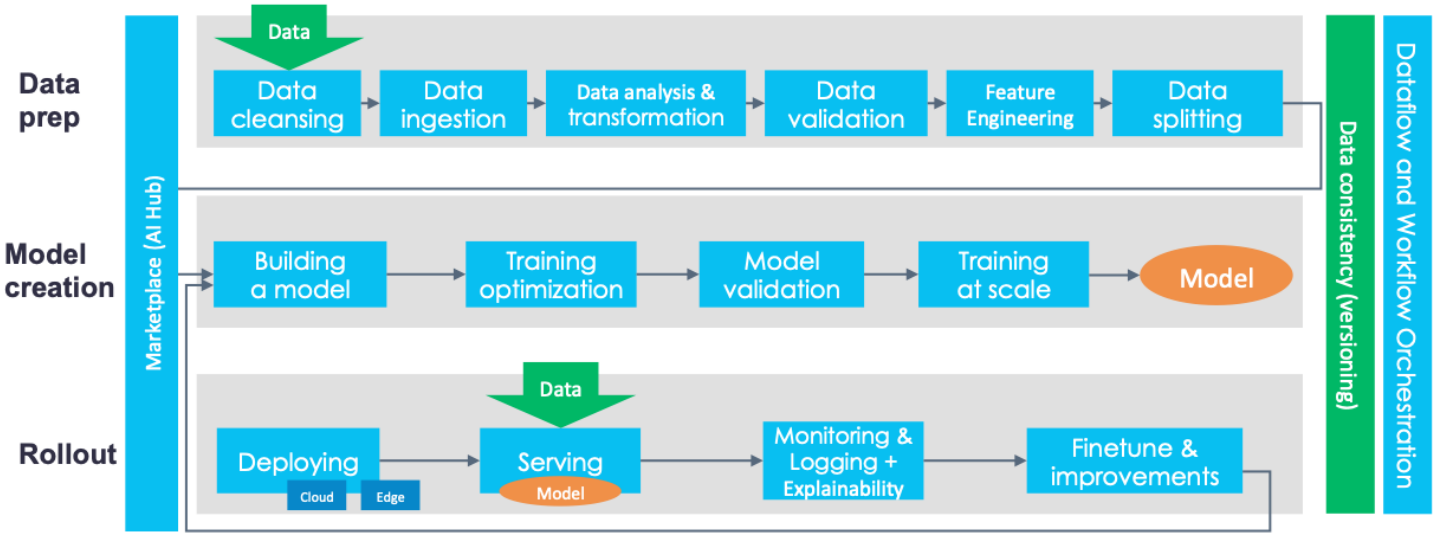
comments

By Ian Hellström, Machine Learning Engineer

Machine Learning (ML) is known as the high-interest credit card of technical debt. It is relatively easy to get started with a model that is good enough for a particular business problem, but to make that model work in a production environment that scales and can deal with messy, changing data semantics and relationships, and evolving schemas in an automated and reliable fashion, that is another matter altogether. If you're interested in learning more about a few well-known ML platforms, you've come to the right place!

As little as 5% of the actual code for machine learning production systems is the model itself. What turns a collection of machine learning solutions into an end-to-end machine learning platform is an architecture that embraces technologies designed to speed up modelling, automate the deployment, and ensure scalability and reliability in production. I talked about lean D/MLOps, data and machine learning operations, before, because machine learning operations without data is pointless, so an end-to-end machine learning platform needs a holistic approach.

The CI/CD Foundation launched an MLOps Special Interest Group (SIG). The steps they have identified for an end-to-end machine learning platform are shown in the next image:



It camouflages a few not-quite-insignificant details, though. For instance, serving may require different technologies depending on whether it's done in real-time or not. Scalable solutions typically have the model inside a container that runs on many machines in a serving cluster that's behind a load balancer. So, a single box in the aforementioned diagram does not imply a single step, container, or component of an actual platform. That's not a critique of the picture, but a warning: what looks simple may not be quite as easy in practice yet.

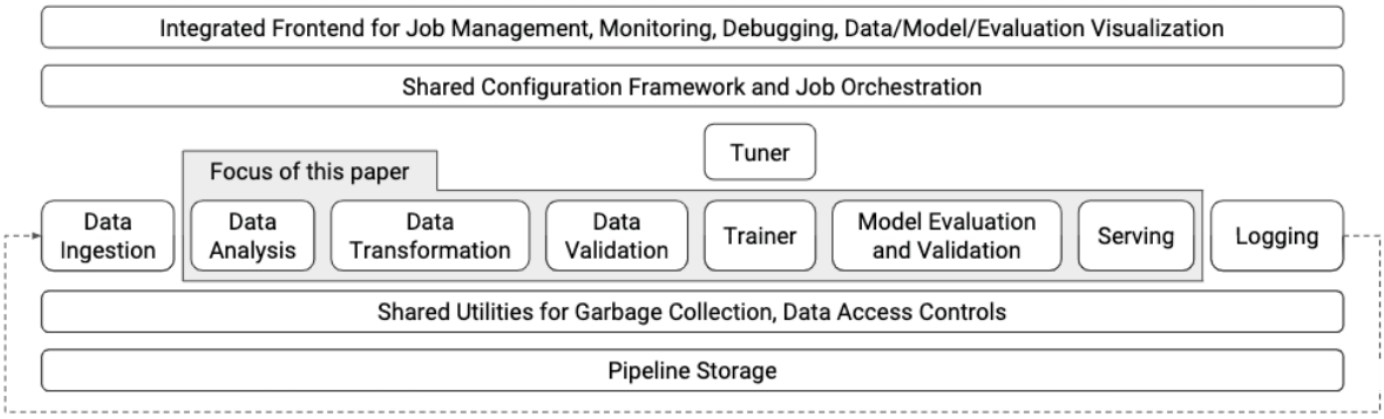
Model (configuration) management is absent from the chart. You can think of things such as versioning, experiment management, run-time statistics, data lineage tracking for training, test, and validation data sets, the capability to retrain a model, either from scratch or incrementally from, say, a snapshot of the model, hyperparameter values, accuracy metrics, and so on.

A crucial aspect that is not listed either is the ability to check the model for bias by, for example, slicing the model's key performance metrics by different dimensions. Many companies need the ability to hot-swap a model or run multiple in parallel, too. The former is important lest a user's request go into the void as it hits the server while the model is updated in the background. And the latter is crucial for A/B testing or model validation.

Another perspective from CI/CD is available here. It mentions the need for versioning data as well as code, which is often overlooked.

Google: TFX

The main motivation behind Google's development of TensorFlow eXtended (TFX) was to reduce the time to productionize a machine learning model from months to weeks. Their engineers and scientists struggled because 'the actual workflow becomes more complex when machine learning needs to be deployed in production.'



TensorFlow and TFX are available freely, although the latter is not as mature as the former, having been released only in 2019, two years after Google presented their ML infrastructure.

Model performance metrics are used to deploy safe-to-serve models. So, if a newer model does not perform as well as an existing one, it is not pushed to production. In TFX parlance, the model does not receive a 'blessing'. With TFX that whole process is automatic.

Here is a quick overview of open-source TFX components:

- ExampleGen ingests and splits the input dataset.
- StatisticsGen calculates statistics for the dataset.
- SchemaGen examines the statistics and creates a data schema.
- ExampleValidator looks for anomalies and missing values in the dataset.
- Transform performs feature engineering on the dataset.
- Trainer trains the model using TensorFlow.
- Evaluator analyses the training results.
- ModelValidator ensures that the model is safe to serve.
- Pusher deploys the model to a serving infrastructure.
- TensorFlow Serving is a C++ backend that serves a TensorFlow SavedModel file.

To minimize training/serving skew, TensorFlow Transform 'freezes' values in the computation graph, so that the same values found during training are used when serving. What may be several operations in the DAG when training will be a single fixed value at serving time.

Uber: Michelangelo

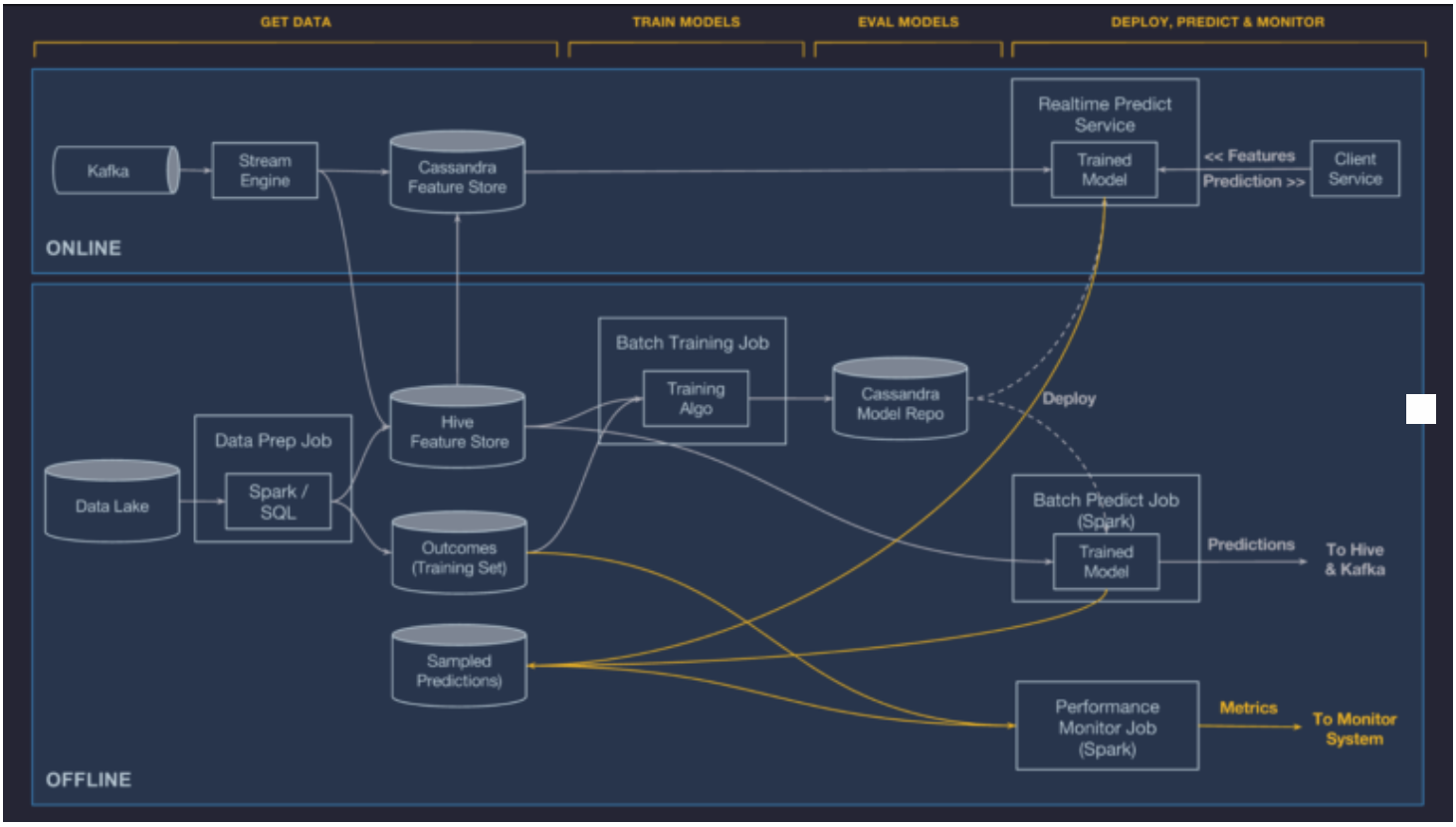
Around 2015, Uber's ML engineers noticed the hidden technical debt in machine learning systems, or the ML equivalent of 'But it works on my machine…' Uber had built custom, one-off systems that integrated with ML models, which was not very scalable in a large engineering organization. In their own words,

> there were no systems in place to build reliable, uniform, and reproducible pipelines for creating and managing training and prediction data at scale.

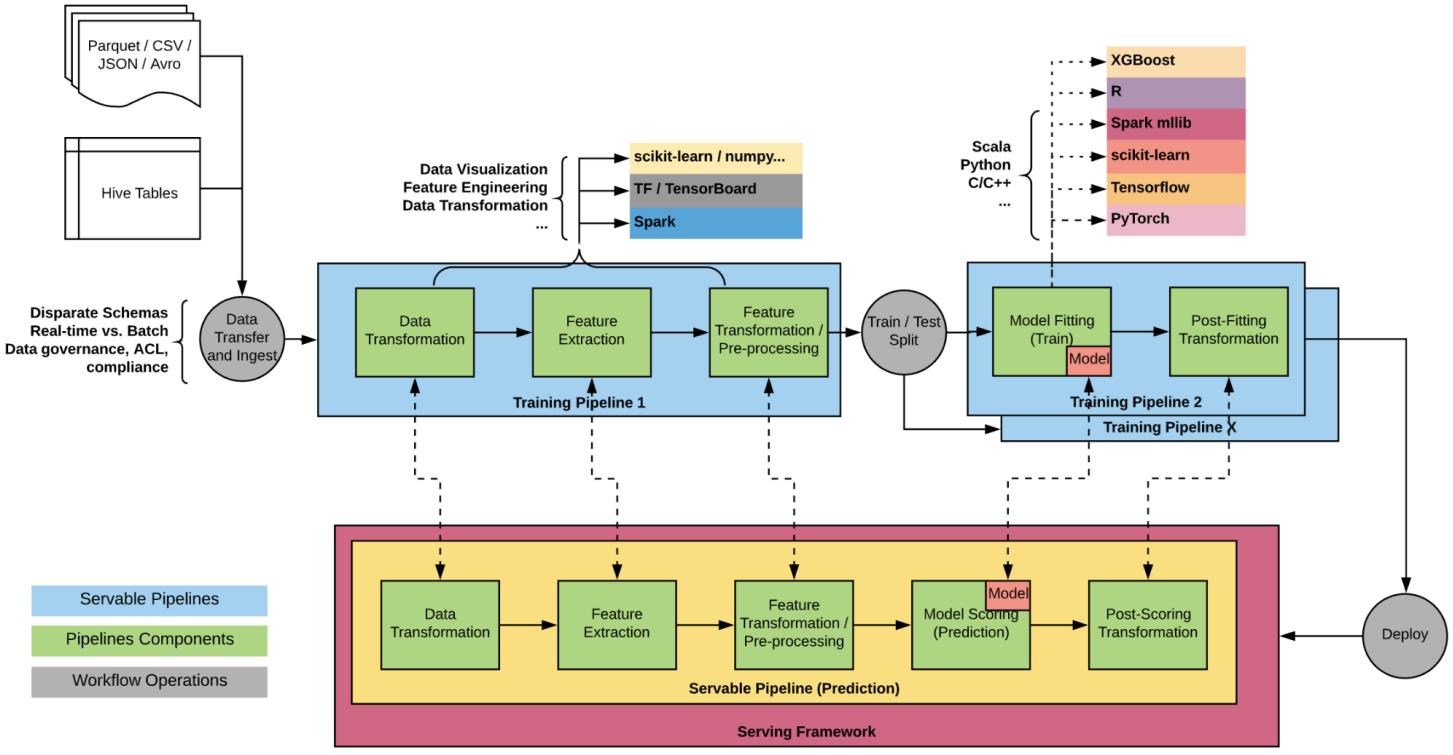
That's why they built Michelangelo. It relies on Uber's data lake of transactional and logged data, and it supports both offline (batch) and online (streaming) predictions. For offline predictions containerized Spark jobs generate batch predictions, whereas for online deployments the model is served in a prediction service cluster, which typically consists of hundreds of machines behind a load balancer, to which clients send individual or batched prediction requests as RPCs.

Metadata relevant to model management (e.g. run-time statistics of the trainer, model configuration, lineage, distribution and relative importance of features, model evaluation metrics, standard evaluation charts, learned parameter values, and summary statistics) are stored for each experiment.

Michelangelo can deploy multiple models in the same serving container, which allows for safe transitions from old to new model versions and side-by-side A/B testing of models.



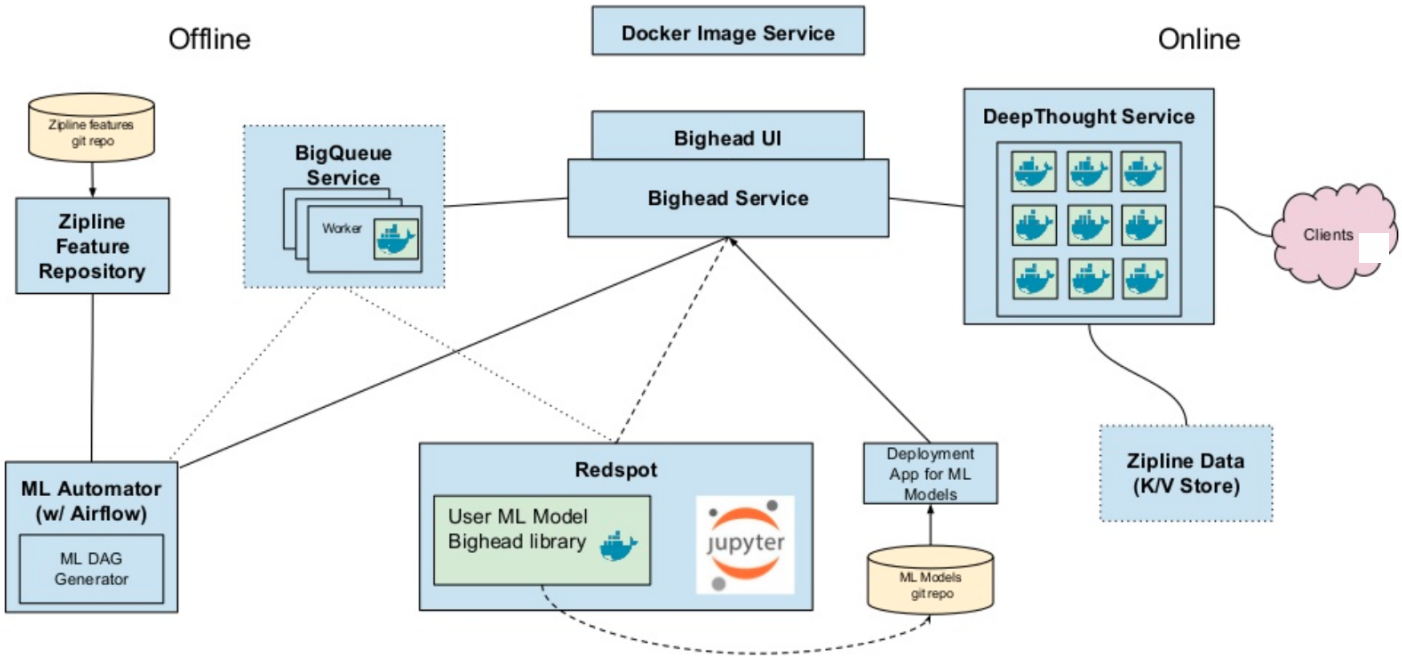
The original incarnation of Michelangelo did not support deep learning's need to train on GPUs, but that the team addressed that omission in the meantime. The current platform uses Spark's ML pipeline serialization but with an additional interface for online serving that adds a single-example (online) scoring method that is both lightweight and capable of handling tight SLAs, for instance, for fraud detection and prevention. It does so by bypassing the overhead of Spark SQL's Catalyst optimizer.



Noteworthy is that both Google and Uber built in-house protocol buffer parsers and representations for serving, avoiding bottlenecks present in the default implementation.

Airbnb: Bighead

Airbnb established their own ML infrastructure team in 2016/2017 for similar reasons. First, they only had a few models in production, but building each model could take up to three months. Second, there was no consistency among models. And third, there were large differences between online and offline predictions. Bighead is the culmination of their efforts:

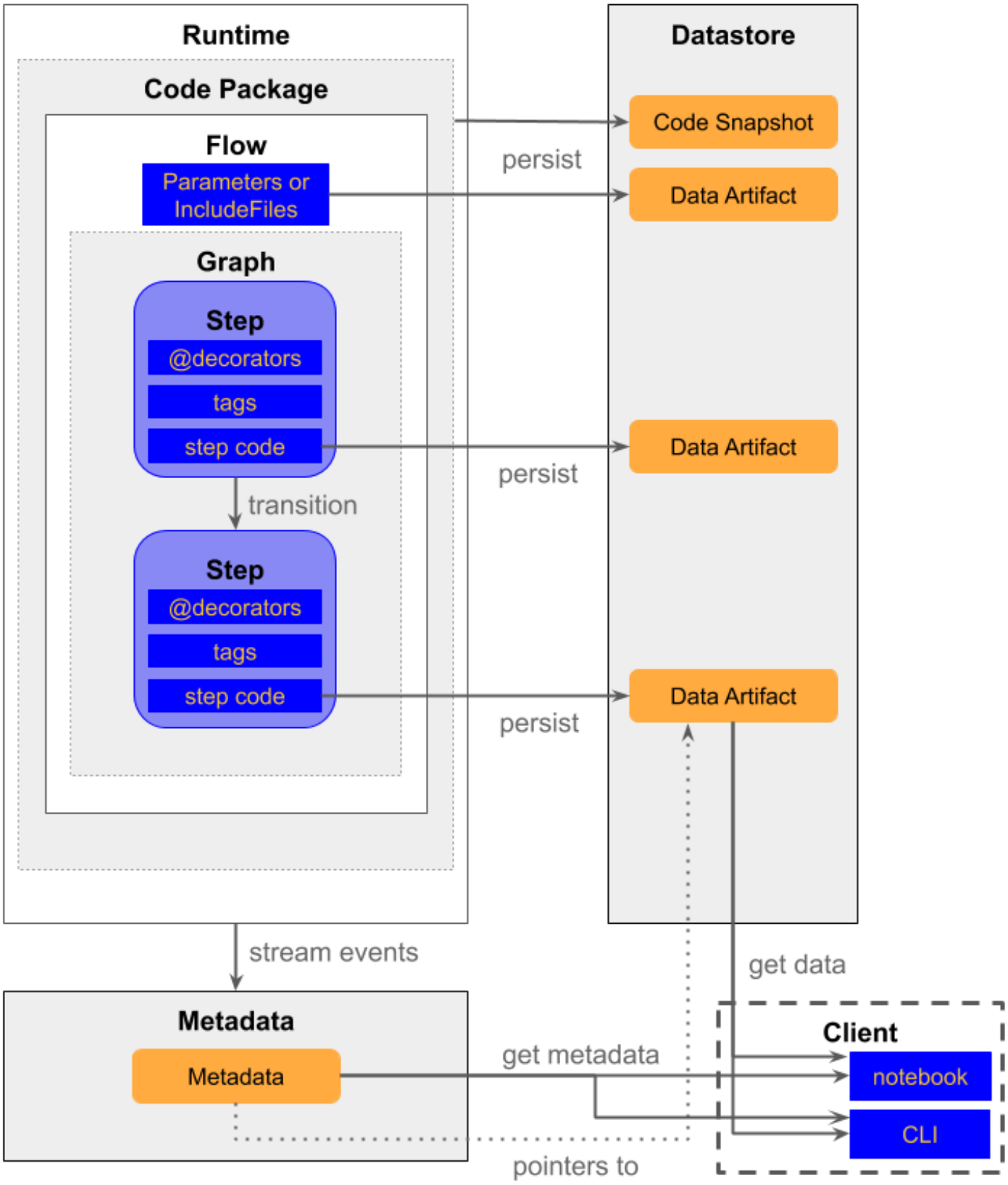


Data management is handled by the in-house tool Zipline. Redspot is a hosted, containerized, multi-tenant Jupyter notebook service. The Bighead library is for data transformations and pipeline abstractions, and it holds wrappers for common model frameworks. It preserves metadata through transformations, so it is used to track lineage.

Deep Thought is a REST API for online predictions. Kubernetes orchestrates the services. For offline predictions, Airbnb use their own Automator.

Netflix: Metaflow

Netflix faced, rather unsurprisingly, similar issues as the aforementioned companies. Their solution was Metaflow, a Python library for data scientists that deals with data management and model training, and not so much prediction serving. As such it is *not* an end-to-end platform for machine learning, and perhaps more geared towards company-internal instead of user-facing use cases. It can of course be turned into a fully-fledged solution with Seldon, which is backed by Kubernetes, or AWS SageMaker. A list of further serving tools is available here.



Data scientists write their workflow as DAG steps, much like data engineers when they use Airflow. And like Airflow, you can use any data science library because to Metaflow it's only Python code that's executed. Metaflow distributes processing and training in the background. All code and data is automatically snapshotted to S3 to ensure there is a version history of each model and experiment. Pickle is the default model serialization format.

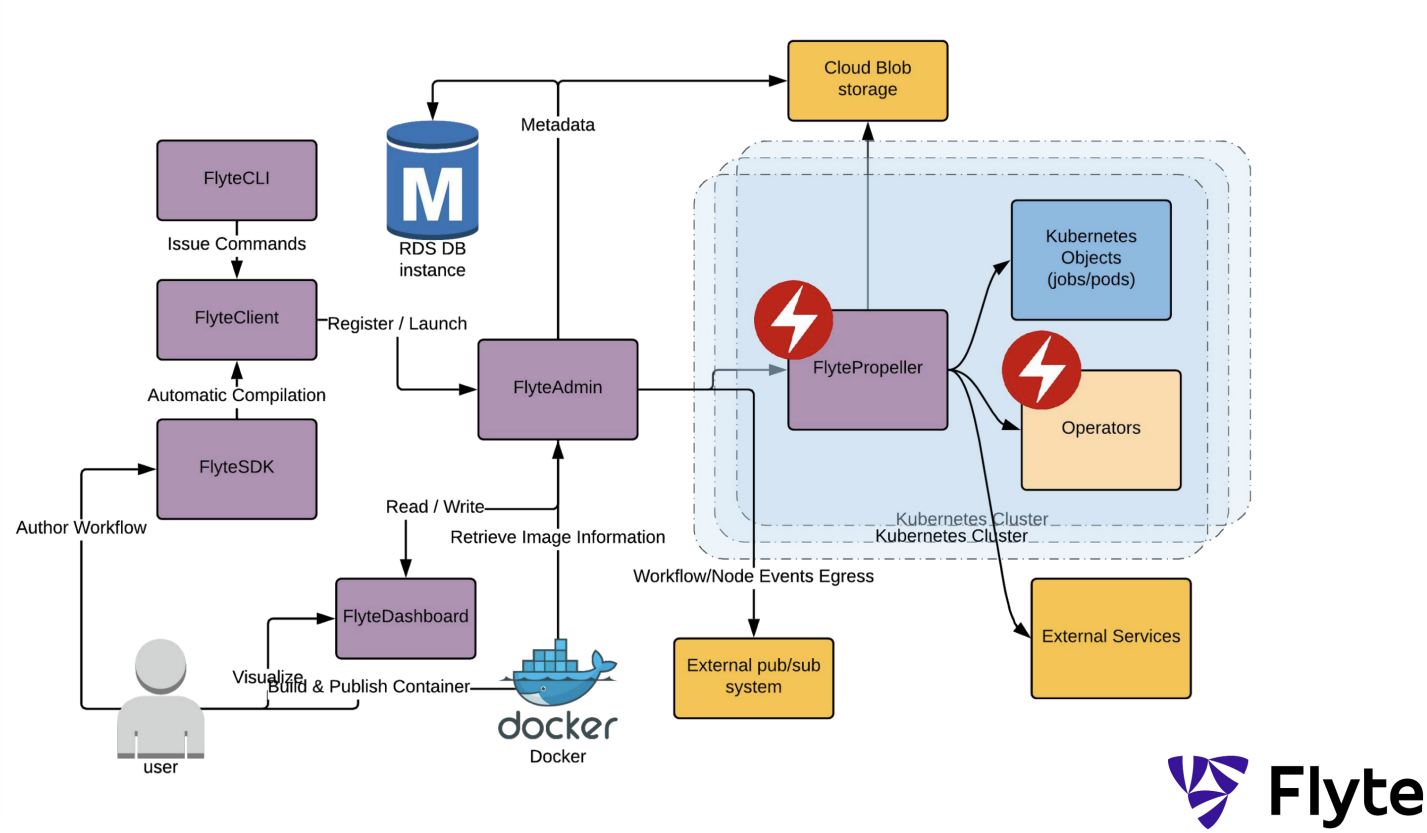
The open-source edition does not yet have a built-in scheduler. It also encourages users to 'primarily rely on vertical scalability', although they can use AWS SageMaker for horizontal scalability. It is tightly coupled to AWS.

Lyft: Flyte

Lyft have open-sourced their cloud-native platform called Flyte, where data and machine learning operations converge. This is in line with my D/MLOps philosophy—Data(Ops) is to MLOps as fuel is to a rocket: without it, ain't nothin' happenin'.

It is built on top of Kubernetes. Since it is used internally by Lyft, it scales to at least 7,000 unique workflows with over 100,000 executions every month, 1 million tasks, and 10 million containers.

All entities in Flyte are immutable, so it is possible to track data lineage, reproduce of experiments, and roll back deployments. Repeated tasks can leverage the task cache to save time and money. Currently supported tasks include Python, Hive, Presto, and Spark as well as sidecars. From looking at the source code it seems EKS is

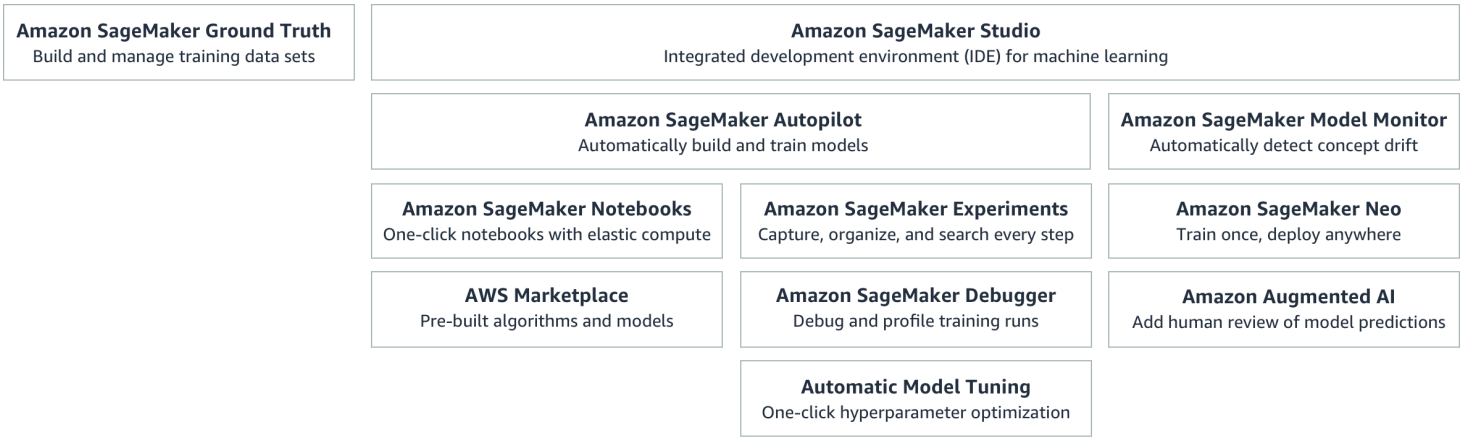


Theirs is also Amundsen, a data catalogue that is not unlike Spotify's Lexikon.

AWS, Azure, GCP, and Co.

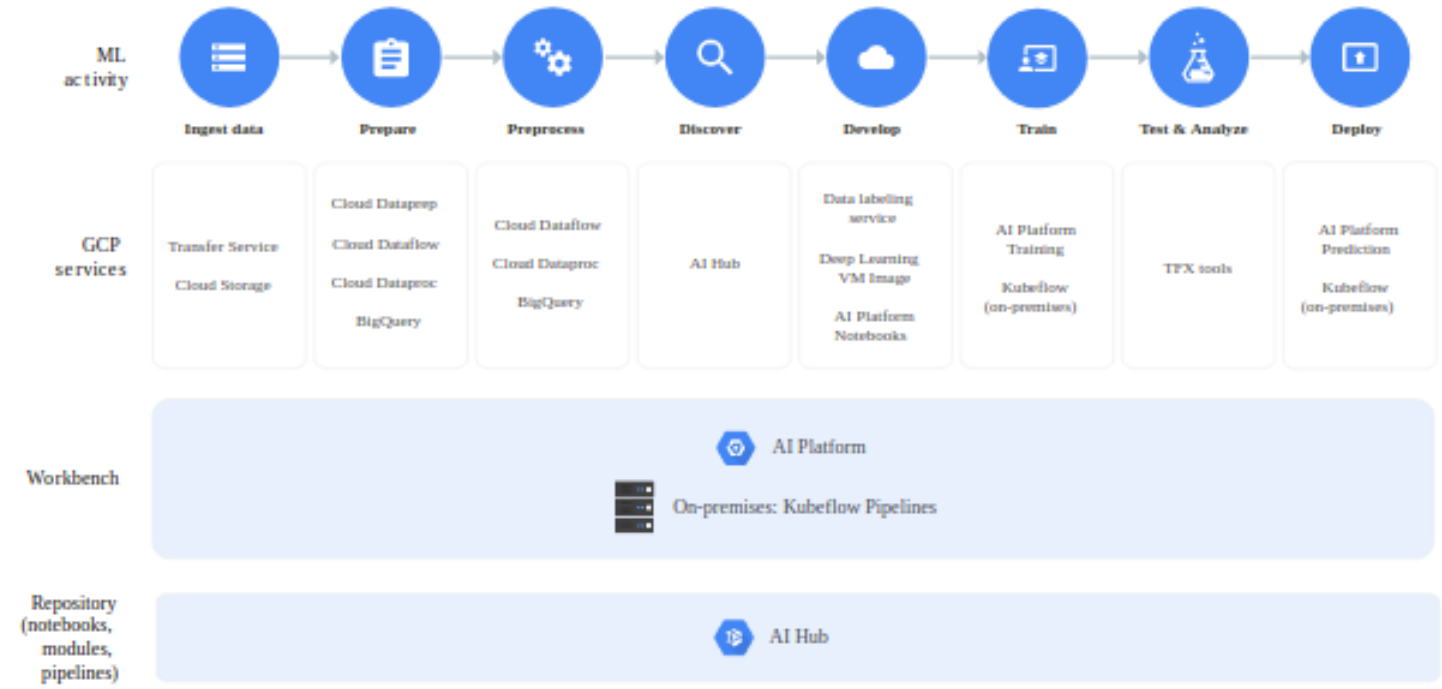
All major players in the public cloud space have their own offerings for machine learning platforms, save for Oracle who only offer canned ML-based models for certain use cases and industries.

AWS SageMaker is a full-stack solution for machine learning that supports TensorFlow, Keras, PyTorch, and MXNet. With SageMaker Neo it's possible to deploy models both in the cloud and on the edge. It has a built-in feature for labelling through Amazon Mechanical Turk for data stored in S3.



Google does not have a managed platform, but with TFX, Kubeflow, and **AI Platform** it's possible to stitch together all the components needed to run models on CPUs, GPUs and TPUs, tune hyperparameters, and automatically deploy to production. Spotify has even opted for the TFX/Kubeflow-on-GCP option.

Beyond TensorFlow, there is support for scikit-learn and XGBoost. Custom containers allow you to use any framework, such as PyTorch. A labelling service à la SageMaker Ground Truth is at the moment in beta.



Azure Machine Learning supports a fair number of frameworks, such as scikit-learn, Keras, PyTorch, XGBoost, TensorFlow, and MXNet. It has its own D/MLOps suite with plenty of graphs. A drag-and-drop interface for model development is available to those who prefer it, but that comes with various caveats. Model and experiment management is done, as expected from Microsoft, with a registry. For production deployments, the Azure Kubernetes Service is used. Controlled roll-outs are possible too.

IBM Watson ML comes with both point-and-click machine learning options (SPSS) and support for a bunch of common frameworks. As the other major players, models are trained on either CPUs or GPUs. Hyperparameter tuning is included in the box too. The platform does not have many details on data and model validation, as these are available in other IBM products.

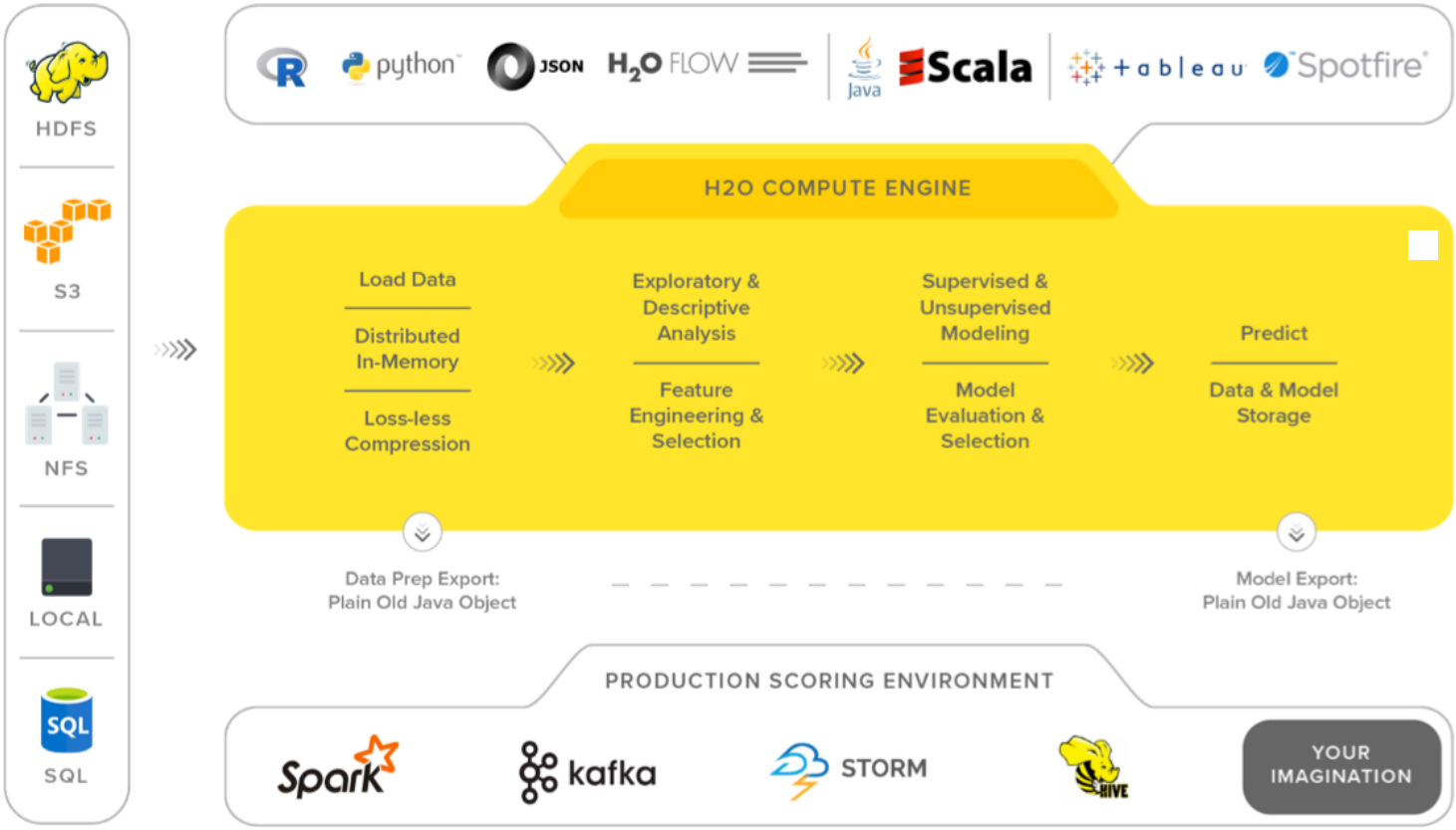
Although **Alibaba's ML Platform for AI** flaunts two buzzwords in one name, it does not improve the documentation; the section on best practices has use cases rather than recommendations.

Anyway, it is heavy on dragging and dropping, especially in data management and modelling, which may not be very conducive to an automated end-to-end ML platform. The platform supports frameworks such as TensorFlow, MXNet, and Caffe, but it also sports a plethora of traditional algorithms. It includes a hyperparameter tuner, as can be expected.

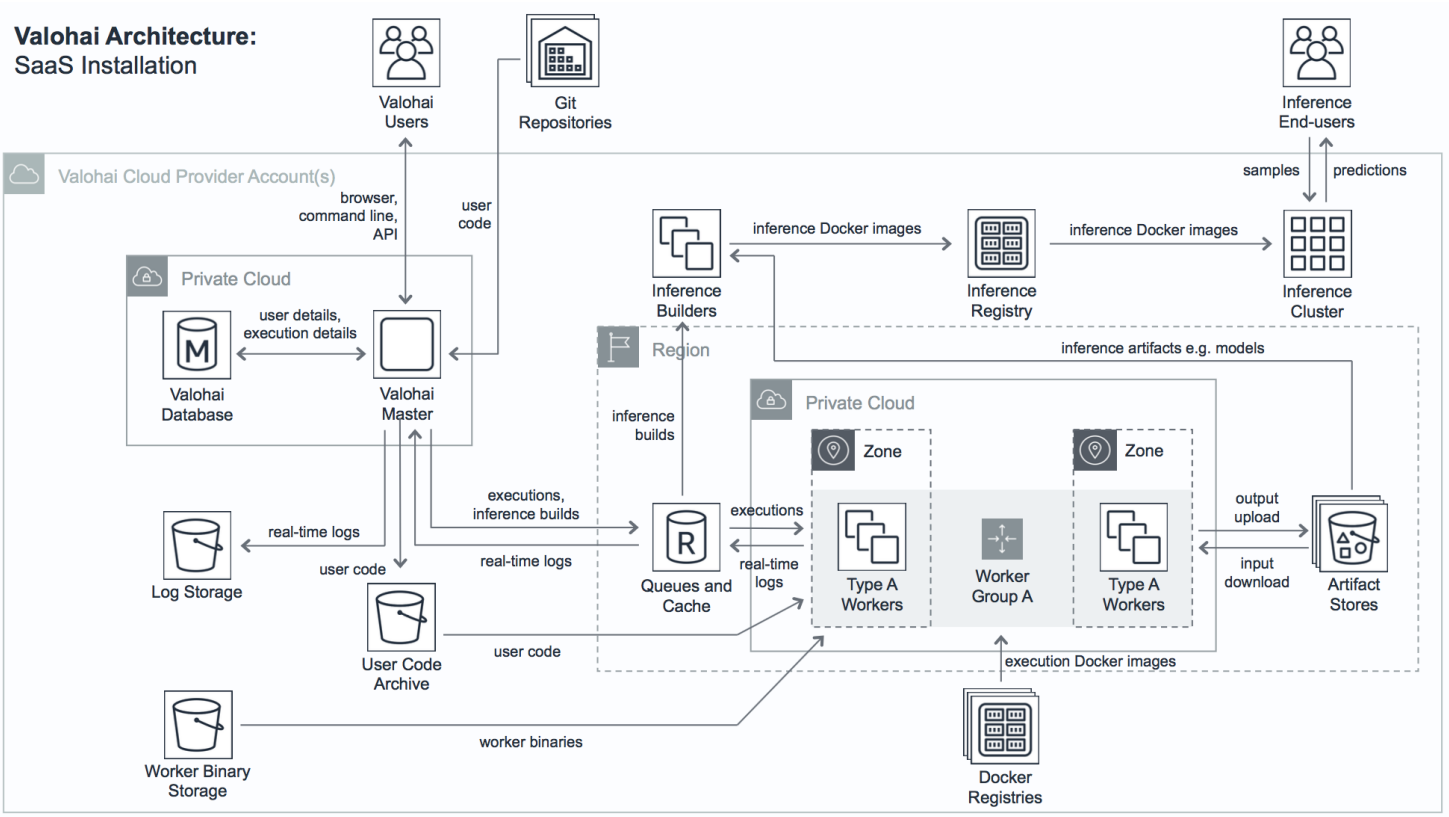
Model serialization is done with PMML, TensorFlow's SavedModel format, or Caffe's format. Please note that a scoring engine that takes a PMML, ONNX, or PFA file may enable quick deployments, but it risks introducing training/serving skew, since the served model is loaded from a different format.

Honourable Mention

H2O offers a platform with data manipulation, various algorithms, cross-validation, grid search for hyperparameter tuning, feature ranking, and model serialization with POJO or MOJO.



Valohai—Finnish for light shark. Really!— is a managed machine learning platform. It can run on private, public, hybrid, or multi-cloud setups.

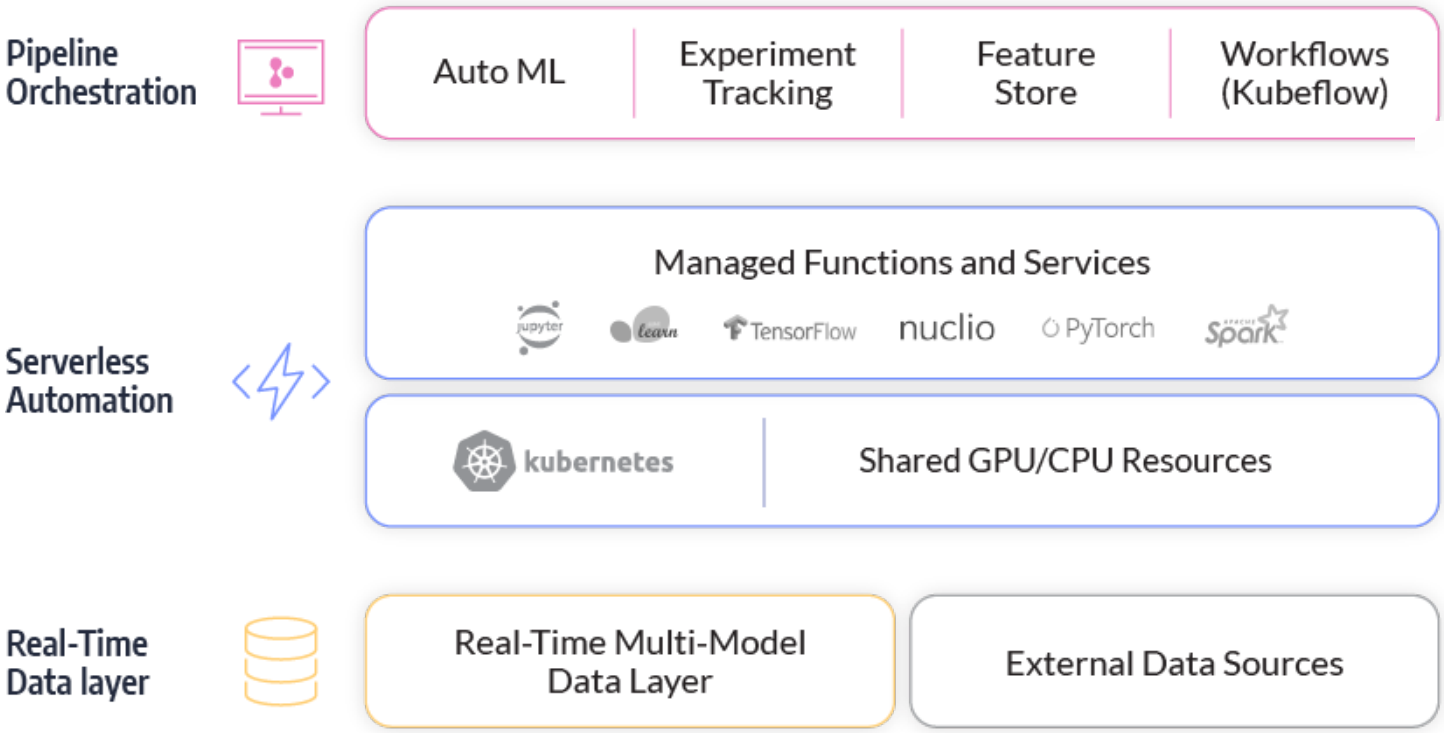


Each operation (or execution) runs a command against a Docker image, so it's very similar to Kubeflow. The main difference is that Valohai manages the Kubernetes deployment cluster for you, whereas Kubeflow requires you to do that. However, Kubeflow and TFX are opinionated in that they provide

some TensorFlow-related tools out of the box. With Valohai you're expected to re-use existing Docker images or roll your own, which means you can use any machine learning framework, but that freedom has to be weighed against maintainability concerns.

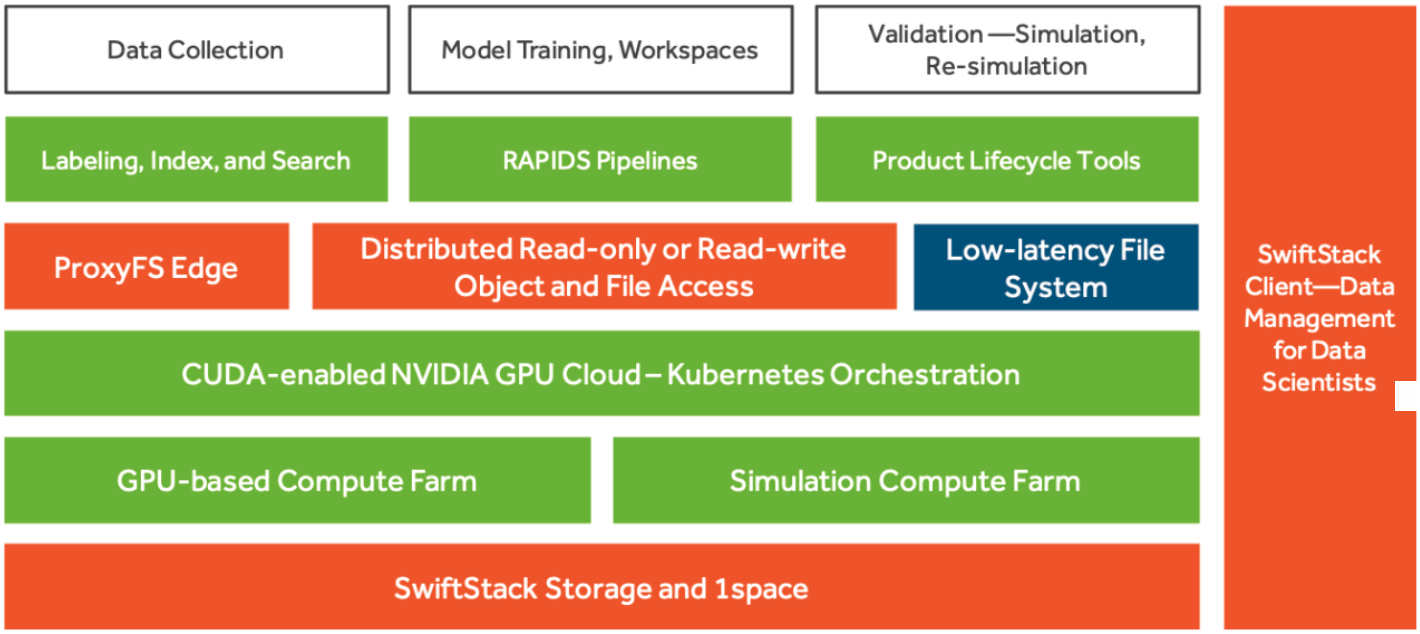
It is therefore possible to distribute training by relying on Spark, Horovod, TensorFlow, or whatever suits your needs and infrastructure best, but it's in your hands to fill in the blanks. It also means you're responsible for ensuring compatibility in data transformations to avoid training/serving skew. Note that it currently only supports object storages.

Iguazio mentions the capability to deploy in seconds from a notebook or IDE, although that seems to miss the most common scenario: a CI/CD pipeline or even the platform itself as with TFX's Pusher component. It uses Kubeflow for workflow orchestration.

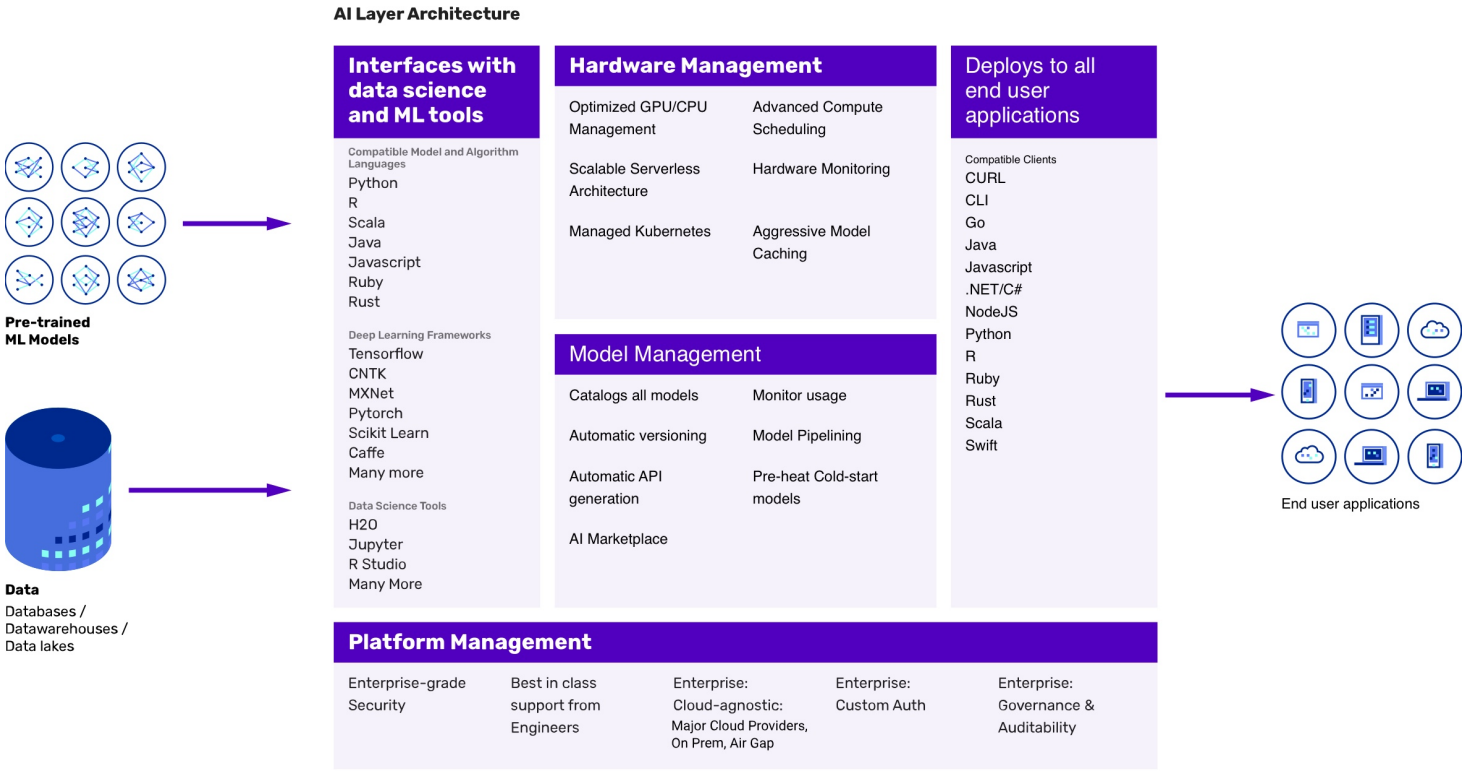


Iguazio does offer a feature store with unified APIs for key-value pairs and time series. Many available products do not come with their own features stores, although most large tech companies have these. A feature store is a central place with ready-to-reuse features that can be shared across models to accelerate model development. It can automate feature engineering on an enterprise scale. From a timestamp, for instance, you can extract many features: year, season, month, day of week, time of day, whether it's a local holiday, elapsed time since last relevant event (recency), how often a certain event happened in a fixed window (frequency), and so on.

SwiftStack AI is geared towards high-throughput deep learning on NVIDIA GPUs with the RAPIDS suite. RAPIDS offers libraries, such as cuML, which allows people to use the familiar scikit-learn API but benefit from GPU acceleration for supported algorithms, and cuGraph for GPU-powered graph analytics.



AI Layer is an API for D/MLOps. It has built-in support for multiple data sources, programming languages, and machine learning frameworks.



MLflow is backed by Databricks, which explains the tight integration with Spark. It offers a limited set of options for deployments. For example, the ability to export a model as a vectorized UDF in PySpark is not the most sensible for real-time systems, since Python UDFs come with the communication overhead between the Python runtime environment and the JVM. The overhead is not as large as with standard PySpark UDFs because Apache Arrow, an in-memory columnar format, is used in the transfer between Python and the JVM, but it's not insignificant. With Spark Streaming as the default data ingestion solution, sub-second latency with Spark's micro-batch model may be tricky to achieve anyway.

Support for logging, which is essential for D/MLOps, is still experimental. From the documentation it follows that MLflow is not focused on data and model validation, at least not as a standard part of the platform itself. There is a managed version of MLflow available (on AWS and Azure) that offers more features.

D2iQ's KUDO for Kubeflow is a Kubeflow-based platform geared towards enterprise customers. Unlike the open-source Kubeflow, it comes with Spark and Horovod as well as pre-built and fully tested CPU/GPU images for the major frameworks: TensorFlow, PyTorch, and MXNet. Data scientists can deploy form within notebooks without the need to switch contexts. By default it supports multi-tenancy. Istio and Dex are integrated for additional security and authentication. KUDO for Kubeflow sits atop Konvoy, D2iQ's managed Kubernetes platform. It can run in the cloud, on-prem, a hybrid, or on the edge. Support for air-gapped clusters is also available.

In Kubernetes-speak, KUDO for Kubeflow is a collection of operators defined with KUDO, a declarative toolkit to create Kubernetes operators using YAML instead of Go. Kubernetes Unified Declarative Operators (KUDOs) for Cassandra, Elastic, Flink, Kafka, Redis, and so on are all open source and

7. **Machine Learning from Scratch: Free Online Textbook**

7. **Free From MIT: Intro to Computer Science and Programming in Python**

Latest News

- Top Stories, Sep 21-27: Introduction to Time Series Ana...
- Looking Inside The Blackbox: How To Trick A Neural Network
- Geographical Plots with Python
- The Online Courses You Must Take to be a Better Data Sc...
- Making Python Programs Blazingly Fast
- Create and Deploy your First Flask App using Python and...

Top Stories
Last Week

Most Popular

1. **Introduction to Time Series Analysis in Python**

2. **Machine Learning from Scratch: Free Online Textbook**
3. **How I Consistently Improve My Machine Learning Models From 80% to Over 90% Accuracy**
4. **New Poll: What Python IDE / Editor you used the most in 2020?**
5. **Automating Every Aspect of Your Python Project**
6. **If I had to start learning Data Science again, how would I do it?**
7. **Making Python Programs Blazingly Fast**

Most Shared

1. **Machine Learning from Scratch: Free Online Textbook**
2. **Introduction to Time Series Analysis in Python**
3. **How I Consistently Improve My Machine Learning Models From 80% to Over 90% Accuracy**
4. **I'm a Data Scientist, Not Just The Tiny Hands that Crunch your Data**
5. **New Poll: What Python IDE / Editor you used the most in 2020?**
6. **The Most Complete Guide to PyTorch for Data Scientists**
7. **What an Argentine Writer and a Hungarian Mathematician Can Teach Us About Machine Learning Overfitting**

More Recent Stories

- Create and Deploy your First Flask App using Python and Heroku
- Causal Inference: The Free eBook
- KDD 2020 Celebrates Recipients of the SIGKDD Best Paper Aw...
- Introduction to Time Series Analysis in Python
- The Most Complete Guide to PyTorch for Data Scientists
- Top tweets, Sep 16-22: An overview of 63 #MachineLearning a...
- How well do you wear your "operationalizing analytics" hat...
- LinkedIn's Pro-ML Architecture Summarizes Best Practices...
- How I Consistently Improve My Machine Learning Models From 80%...
- Artificial Intelligence for Precision Medicine and Better Heal...
- KDnuggets 20:n36, Sep 23: New Poll: What Python IDE / Edito...
- MathWorks Deep learning workflow: tips, tricks, and often forg...
- New Poll: What Python IDE / Editor you used the most in 2020?
- Machine Learning from Scratch: Free Online Textbook
- The Potential of Predictive Analytics in Labor Industries
- Statistical and Visual Exploratory Data Analysis with One Line...
- I'm a Data Scientist, Not Just The Tiny Hands that Crunch yo...
- Top Stories, Sep 14-20: Automating Every Aspect of Your Python...
- What an Argentine Writer and a Hungarian Mathematician Can Tea...
- Automating Every Aspect of Your Python Project

can be integrated with the platform. More details are described in an introductory article by yours truly.

If you want to see yet more options, including visual workbenches, have a look here or check out Gartner's magic quadrant for data science and machine learning platforms. Facebook have also published details of their platform FBLearner Flow (2016), as well as LinkedIn (2018) and eBay (2019).

Bio: Ian Hellström has been a data and machine learning engineer at various companies, including D2iQ, Spotify, Bosch and Sievo. He is the product manager for D2iQ's enterprise machine learning platform KUDO for Kubeflow. He currently resides in Hamburg, Germany.

Original. Reposted with permission.

Related:

- How to Extend Scikit-learn and Bring Sanity to Your Machine Learning Workflow
- A Layman's Guide to Data Science. Part 3: Data Science Workflow
- Deploy a Machine Learning Pipeline to the Cloud Using a Docker Container

<= Previous post
Next post =>

Top Stories Past 30 Days

Most Popular

1. If I had to start learning Data Science again, how would I do it?
2. Free From MIT: Intro to Computer Science and Programming in Python
3. Automating Every Aspect of Your Python Project
4. Top Online Masters in Analytics, Business Analytics, Data Science Updated
5. Introduction to Time Series Analysis in Python
6. Autograd: The Best Machine Learning Library Youre Not Using?

Most Shared

1. Modern Data Science Skills: 8 Categories, Core Skills, and Hot Skills
2. Top Online Masters in Analytics, Business Analytics, Data Science – Updated
3. Machine Learning from Scratch: Free Online Textbook
4. How to Evaluate the Performance of Your Machine Learning Model
5. Deep Learning's Most Important Ideas
6. Statistics with Julia: The Free eBook