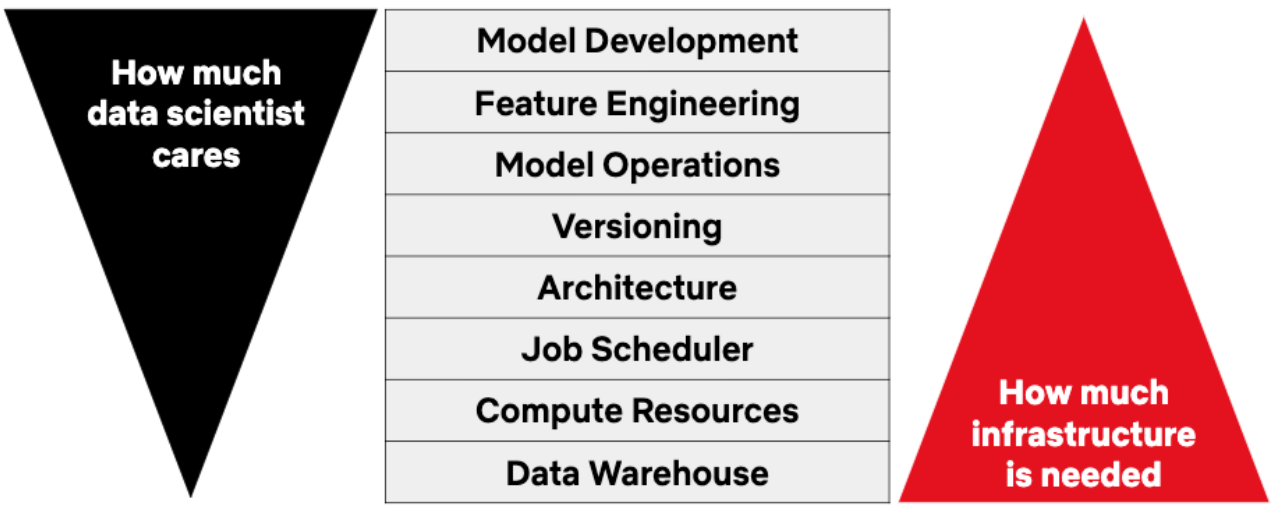
NOTES FROM INDUSTRY

Lessons on ML Platforms — from Netflix, DoorDash, Spotify, and more

Ernest Chan May 12 · 12 min read

Your data scientists produce wonderful models, but they can only deliver value once the models are integrated into your production systems. How do you make it easy for the data scientists to *repeatedly* deliver value? What do you build, what do you buy, and what tools do you need to solve your organization's problems specifically?

Many companies who do ML well have invested in ML platforms to enable data scientists to deliver value quicker. These platforms solve problems common to many ML tasks and empower data scientists to focus on where they can uniquely add value. The goal of platforms is aptly summarized by this diagram from the Metaflow project.



Source: https://docs.metaflow.org/introduction/what-is-metaflow

We can learn from the ML platforms of successful tech companies to inform what we build for our organization. As Bruce Lee put it:

> *"Absorb what is useful, discard what is useless and add what is specifically your own"*

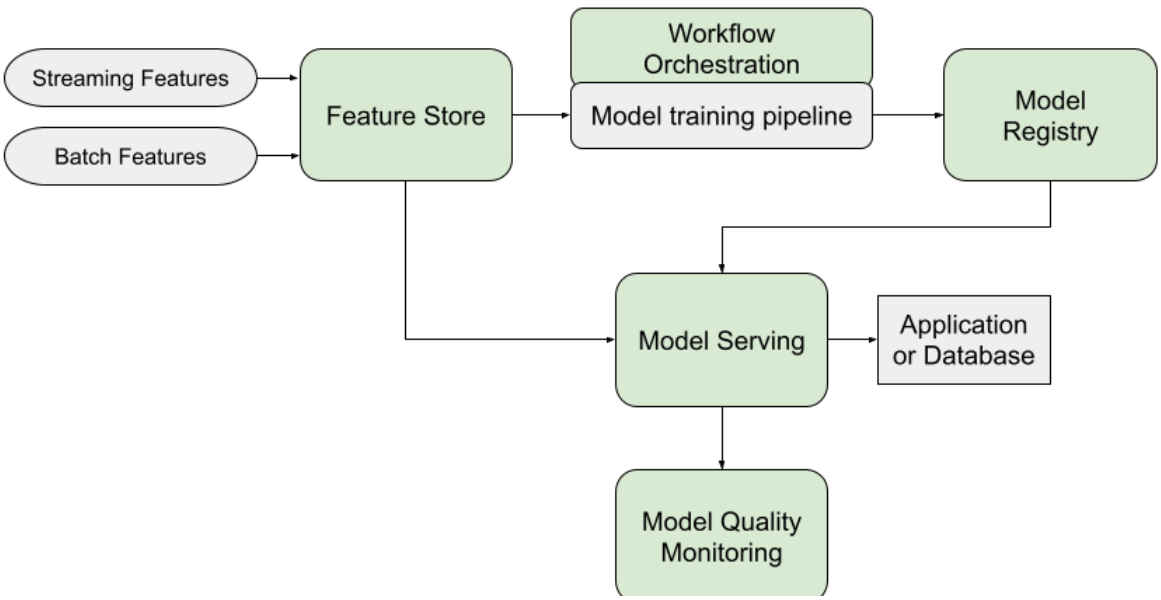
Through scouring conference talks and blog posts from the past several years, I've documented ML platforms' common components and capabilities at eleven large tech companies.

This post contains:

- A high-level overview of common ML platform components
- A table of tools used by each company
- Observations about the components
- The platform user experience
- A summary of capabilities unique to certain companies

Common ML platform Components

At a high level, five ML platform components stand out which are indicated by the green boxes in the diagram below.



Common components in an ML platform. Diagram by the author

Common components explained:

- Feature store: A service that stores batch and streaming features and serves features for both offline and online use. A feature store usually supports low-latency serving, guarantees consistency between offline and online features, and allows both contribution and access to features across teams. If you want to learn more about feature stores, read Feature Stores — A Hierarchy of Needs by Eugene Yan.

- Workflow orchestration: Model training jobs are oftentimes represented as a directed acyclic graph (DAG) and need a tool that orchestrates the execution of tasks. DAGs may range from a simple two-step DAG of (model training) -> (model validation) to complex structures involving hyper-parameter tuning and multiple models. I won't go into the model training pipeline as a component because typically, the platform handles orchestration, and what happens in model training is up to the data scientists.

- Model registry: Trained models get stored here. Model metadata about the training set, hyper-parameters, performance metrics, and so on are also recorded. Normally there's a UI for users to inspect their models.

- Model serving: This system retrieves models from the model registry, fetches features from the feature store, and outputs predictions. Predictions are sent to another production system (like a service for product recommendations) or get stored in a database in the case of batch scoring. Alternatively, sometimes models are embedded in production services to minimize latency and there isn't a separate model serving service.

- Model quality monitoring: A model's life begins once it's deployed. To ensure it's living a purposeful life you need to monitor its performance on live data. Not shown in the diagram, the monitoring system may feed metrics back to the model registry or some UI, and connect to an alerting system.

Tools Used by Each Company

The table below illustrates each company's tool for each of the five components described above. If a cell is missing, it doesn't mean they don't have that component, only that I couldn't find information about it.

IH means they built the component in-house and the named tools are open-source.

	Model registry	Feature Store	Workflow Orchestration	Model serving	Model quality monitoring
Netflix	IH	Kind of*	Metaflow	IH	IH
Intuit	IH	IH	Argo Workflows	IH	IH
Intel	mlflow			Seldon Core	IH
Booking.com	IH	IH		IH	IH
Stitch Fix	IH	IH	IH	IH	
DoorDash	IH	IH		IH	IH
Uber	IH	IH	IH	IH	IH
Paypal	IH	IH	Airflow	IH	
Spotify	TFX ML Metadata	IH	Kubeflow Pipelines	IH	TF Model Analysis
Etsy		IH	Airflow	IH	IH
Pinterest	mlflow	IH	Airflow fork	IH	IH

Tools used for common ML platform components, for each of the eleven companies. Table by author.

* Netflix uses feature encoders, which is transformation code that can be invoked at both training and scoring time.

Caveat: This is a biased sample. Most of these companies are large, B2C, based in the U.S., and have numerous opportunities for ML solutions. Their requirements for ML systems may not match your organization's. This is all to say you may not need all the components they have.

Notes on the Above Table

Everyone's Got a Model Registry

Why? You need to know what gets deployed into your production systems. Like image registries for code artifacts, model registries manage your model artifacts. Models

require a dedicated system because their behavior is determined not only by code, but also by the training data, and hyper-parameters. These three aspects should be linked to the artifact, along with metrics about performance on hold-out data. All this information is useful for reproducibility, debugging, and to inform model improvements.

2021: The Year of Feature Store FOMO

The CEO of Tecton, a feature store company, calls 2021 the year of the feature store. For a lot of us, it's the year of feature store FOMO, as shown in the table above. Almost every company has a feature store. It makes sense they're all built in-house because there weren't many commercial or open-source feature store offerings until recently. More importantly, custom-built feature stores can be tailored to specific requirements, which for these companies includes large-scale production traffic.

Building a feature store is not a walk in the park. This series of posts provides a tour of feature stores and insights into when an organization needs one. One criterion for needing a feature store is your models need data from a large number of data sources. I recommend you go read the post for the other factors. Just because it seems like everyone else has a feature store doesn't mean you need one!

Workflow Orchestration

This column shows the highest usage of open source tools. Workflow orchestration tools abound on Github, among which are mature and battle-tested frameworks like Airflow and Argo. Chances are, one of these tools will suit your company's needs.

Model serving

Most serving systems are built in-house, I assume for similar reasons as a feature store — there weren't many serving tools until recently and these companies have stringent production requirements.

A common pattern for model serving is to provide multiple paved paths for model deployment, each with different tradeoffs.

Booking.com considers a 2D plane of flexibility vs robustness, and each of their four deployment options makes different trade-offs in that plane. A robust but non-flexible method is to precompute model outputs for all possible inputs, and store predictions in a fast lookup table. This option is non-flexible because it only works if the space of inputs is small and it's acceptable to provide batch predictions. But, it's very performant, and serving the model is very simple.

Pinterest supports three options. One is to bring a service that provides model predictions. The second is to embed the model in an existing production service. The third is to deploy the model on a platform-managed service that handles feature retrieval, caching and supports a large number of requests per second.

At Uber, users can deploy Python-based models, which are more suitable for smaller-scale applications or to quickly try out a new idea. This option is more accessible to most data scientists. Alternatively, users can use the JVM-based serving system for larger-scale demands.

Model Quality Monitoring

Problems like training-serving skew, concept drift, and upstream data issues can cause degradations in customer experience without throwing off any production alarms. By monitoring model quality you can reduce incidents of silent failures and provide timely remediations.

Some model quality monitoring capabilities provided by the platforms:

- Track the distributions of model inputs and outputs over time. DoorDash, Booking.com, and Uber all have this capability. When predictions go awry it's easier to determine whether it's a data issue, and which input(s) caused the issue by visualizing changing distributions. This capability also enables alerts on feature distribution shifts.
- Concept drift detection. Drift may trigger model retraining over a fresher dataset.
- Compute model predictive performance when labels become available. Some platforms provide computation and monitoring for predefined metrics like AUC and precision/recall.

While monitoring for operational metrics like latency and queries per second may be agnostic to the ML problem, monitoring for model quality is usually context-specific. Many platforms provide built-in monitoring for the former and customizable tooling for the latter. At Netflix, data scientists can schedule their own notebooks to monitor deployed models. Intuit has a service that allows data scientists to define monitoring pipelines through config files.

The Platform User Experience

This section covers the experience of platform users, who tend to be data scientists or ML engineers.

How users interact with the platform

Users tend to interact with components via:

- User Interfaces
- Config files
- APIs/Libraries

Slick **user interfaces** provide capabilities like one-click model deployment, a view into model performance and input features, model version tracking, and data lineage tracking. Companies like Netflix and Uber provide very comprehensive UIs, but that's not the norm due to the engineering effort required.

Config files simplify certain tasks. For example, Intuit's platform users can define model training jobs in a YAML config and specify parameters like the training data source, validation data source, and compute resources. Intel's platform allows users to define model inference graphs in YAML as an abstraction over Seldon Core.

A notable use of **APIs/Libraries** is in simplifying feature access. Common across multiple feature stores is the ability to switch from batch to online features with a single line change. E.g. `client.get_batch_features` to `client.get_online_features`. As another example of libraries, Spotify has a CLI that helps users build Docker images for Kubeflow Pipelines components. Users rarely need to write Docker files.

Data scientists own the ML problem end-to-end

A common goal of an ML platform is to let data scientists own the ML problem from research to production. Ideally, data scientists don't hand off the model to an engineer for reimplementation. When data scientists can stay close to the problem, they can more easily solve modeling issues or devise modeling improvements.

Python Model Support

Python is the lingua franca of data science and companies eventually realize they need to provide a road for deploying Python models. Uber and Booking.com's ecosystem was originally JVM-based but they expanded to support Python models/scripts. Spotify made heavy use of Scala in the first iteration of their platform until they received feedback like:

> *some ML engineers would never consider adding Scala to their Python-based workflow.*

Unique Platform Capabilities

This section covers platform capabilities unique to one or two companies within the eleven.

Easy Performance Testing and Compute Resource Suggestions

Data scientists don't necessarily know what resource configurations the model server needs to meet SLAs. Intuit's platform allows users to define the requirements of their deployed models, such as requests per second and latency. The platform will then try various resource configurations such as compute instance types and suggest a setup that meets the requirements.

Track Downstream Consumers

The paper "Hidden Technical Debt in Machine Learning Systems" from Google illustrates the problems with undeclared downstream consumers — systems that consume your model's outputs without telling you. To quote:

> *they create a hidden tight coupling of model m_a to other parts of the stack. Changes to m_a will very likely impact these other parts, potentially in ways that are unintended, poorly understood, and detrimental.*

A related issue in software engineering is API contract changes. Once your service has multiple consumers you have to be very careful not to change your API in backward-incompatible ways. For ML models, what constitutes a "backward-incompatible" data change is not as clear. A subtle change in the definition of an input (say when nulls occur) can cause unexpected behavior.

Netflix's model management platform, Runway, allows users to find all deployed models that use a certain input feature and all upstream models that provide inputs to the chosen model. Users can also see what services consume a deployed model's outputs. All consumers are automatically declared, and users can understand what gets affected by a modeling change.

Encourage Idempotency

Stitch Fix's talk emphasized the importance of idempotent jobs — you should be able to run the same job multiple times and get the same result. With idempotency, you can retry pipelines without worrying about unintended side effects. Stitch fix implemented guard rails, like code templates, that encourage idempotent logic.

Log inputs and outputs of your online models

If you've ever learned a phrase in a new language but failed to comprehend when a native speaker said the same phrase, you've encountered training-serving skew. Your listening comprehension wasn't trained on how native speakers actually speak.

In ML, models that seem strong based on offline metrics can be rendered useless in an online setting due to training-serving skew. To prevent this skew, companies like DoorDash and Etsy log a variety of data at *online* prediction time, like model input features, model outputs, and data points from relevant production systems.

This approach ensures point-in-time correctness of the logged data. Data scientists can train models on this data and be confident in the test set results. However, a trade-off is it's hard to ensure correctness for data not captured in the online setting. A data scientist may want to experiment with a new feature that relies on non-captured data. They'll need to create a new feature definition for the online capture system and wait several months to accumulate enough data for model training.

Model composition

At Paypal, a major use case for ML is fraud prevention, which requires the flexible composition of models. For example, one model may detect anomalous IP addresses for the user, another may detect a specific type of merchant fraud, and multiple specialized models are combined to output a fraud determination.

Paypal's inference engine supports the flexible composition of models. For instance, model A feeds into model B, and depending on the output, models C or D is invoked. Another scheme is to run different models on the same data to compare performance. Seldon Core, used by Intel, provides similar model composition capabilities.

Conclusion

This post covered a wide range of ML platform topics — from common ML components to prevalent practices to unique capabilities. I hope it aids your ML system development journey. If you'd like more in-depth writing on a particular type of component, please let me know!

References

- Baer, Josh, and Samuel Ngahane. "The Winding Road to Better Machine Learning Infrastructure Through Tensorflow Extended and Kubeflow." Spotify Engineering, Spotify, 6 July 2020, engineering.atspotify.com/2019/12/13/the-winding-road-to-

better-machine-learning-infrastructure-through-tensorflow-extended-and-kubeflow.

- Bernardi, Lucas. “Machine Learning in Production: The Booking.Com Approach.” Medium, 18 Nov. 2019, booking.ai/https-booking-ai-machine-learning-production-3ee8fe943c70.
- Canchi, Srivathsan, and Tobias Wenzel. “OpML ’20 — Managing ML Models @ Scale — Intuit’s ML Platform.” YouTube, uploaded by USENIX, 17 July 2020, www.youtube.com/watch?v=OVysmLWo3pM&feature=emb_title.
- Chintalapani, Sriharsha, and Sandeep Karmakar. “No Code Workflow Orchestrator for Building Batch & Streaming Pipelines at Scale.” Uber Engineering Blog, 11 Dec. 2020, eng.uber.com/no-code-workflow-orchestrator.
- Falsina, Luca, and Brammert Ottens. “Scaling Machine Learning at Booking.Com with H2O Sparkling Water and FeatureStore.” Vimeo, uploaded by Databricks, 6 June 2018, vimeo.com/274397355.
- Fania, Moty. “Enabling Push Button Productization of AI Models.” YouTube, uploaded by Databricks, 10 July 2020, www.youtube.com/watch?v=GhatQC6o3J8.
- Hermann, Jeremy. “Michelangelo — Machine Learning @Uber.” InfoQ, uploaded by InfoQ, 23 Mar. 2019, www.infoq.com/presentations/uber-ml-michelangelo.
- Hu, Sheila, and Aakash Sabharwal. “Apply() Conference 2021 | Towards a Unified Real-Time ML Data Pipeline, from Training to Serving.” YouTube, uploaded by Tecton, 11 May 2021, www.youtube.com/watch?v=s1DyAppdNmQ&t=214s.
- Khan, Aman. “Apply() Conference 2021 | Feature Stores at Spotify: Building & Scaling a Centralized Platform.” YouTube, uploaded by Tecton, 10 May 2021, www.youtube.com/watch?v=mItriAtSrgs.
- Kourjanski, Mikhail. “ML Data Pipelines for Real-Time Fraud Prevention @PayPal.” InfoQ, uploaded by InfoQ, 22 Aug. 2018, www.infoq.com/presentations/paypal-ml-fraud-prevention-2018.
- Krawczyk, Stefan. “‘Deployment for Free’: Removing the Need to Write Model Deployment Code at Stitch Fix.” Slideshare, 29 Apr. 2021,

www.slideshare.net/StefanKrawczyk/deployment-for-free-removing-the-need-to-write-model-deployment-code-at-stitch-fix.

- Li, Yulei, et al. “Airflow as the next Gen of Workflow System at Pinterest.” YouTube, uploaded by Apache Airflow, 24 July 2020, www.youtube.com/watch?v=KpCPfooD5hM.
- Liu, David. “Apply() Conference 2021 | Evolution and Unification of Pinterest ML Platform.” YouTube, uploaded by Tecton, 7 May 2021, www.youtube.com/watch?v=8Swp9xM-rLY.
- Luu, Hien, and Arbaz Khan. “Apply() Conference 2021 | Scaling Online ML Predictions to Meet DoorDash Growth.” YouTube, uploaded by Tecton, 5 May 2021, www.youtube.com/watch?v=_iipJI4HKf0.
- Lyan, Marina. “ML-Centric Software Engineering at PayPal | PayPal AI.” Medium, 30 Jan. 2021, medium.com/paypal-ai/ml-centric-software-engineering-83a97331488c.
- Magnusson, Jeff. “Developing Data and ML Pipelines at Stitch Fix.” InfoQ, uploaded by InfoQ, 15 May 2018, www.infoq.com/presentations/data-ml-pipelines-stitchfix.
- McHale, Kevin, and Kevin McHale. “Boundary-Layer : Declarative Airflow Workflows.” Code as Craft, 14 Nov. 2018, codeascraft.com/2018/11/14/boundary-layer%E2%80%89-declarative-airflow-workflows.
- Mehta, Romit. “PayPal Notebooks, Powered by Jupyter: Enabling the next Generation of Data Scientists at Scale.” Medium, 19 Sept. 2018, medium.com/paypal-tech/paypal-notebooks-powered-by-jupyter-fd0067bd00b0.
- Peng, Liping, and Eugen Cepoi. “OpML ’20 — Runway — Model Lifecycle Management at Netflix.” YouTube, uploaded by USENIX, 17 July 2020, www.youtube.com/watch?v=kvl4lCIMqio&feature=emb_title.
- Ramesh, Raghav. “Engineering Systems for Real-Time Predictions @DoorDash.” InfoQ, uploaded by InfoQ, 22 Aug. 2018, www.infoq.com/presentations/doordash-real-time-predictions.

- Tuulos, Ville. “Human-Centric Machine Learning Infrastructure @Netflix.” InfoQ, uploaded by InfoQ, 19 Dec. 2018, www.infoq.com/presentations/netflix-ml-infrastructure.

- Venkatasubbaiah, Sumanth, and Qingbo Hu. “How Intuit Uses Apache Spark to Monitor In-Production Machine Learning Models at Large-Scale.” YouTube, uploaded by Databricks, 21 Aug. 2020, www.youtube.com/watch?v=woVFk1Imvu8&t=7s.

- Sculley, D., Holt, Gary, Golovin, Daniel, Davydov, Eugene, Phillips, Todd, Ebner, Dietmar, Chaudhary, Vinay, Young, Michael, Crespo, Jean-Francois and Dennison, Dan. “Hidden Technical Debt in Machine Learning Systems.” Paper presented at the meeting of the 28th International Conference on Neural Information Processing Systems (NIPS), 2015.

Machine Learning | Machine Learning Platform | Software Architecture | Data Science Tools | Notes From Industry