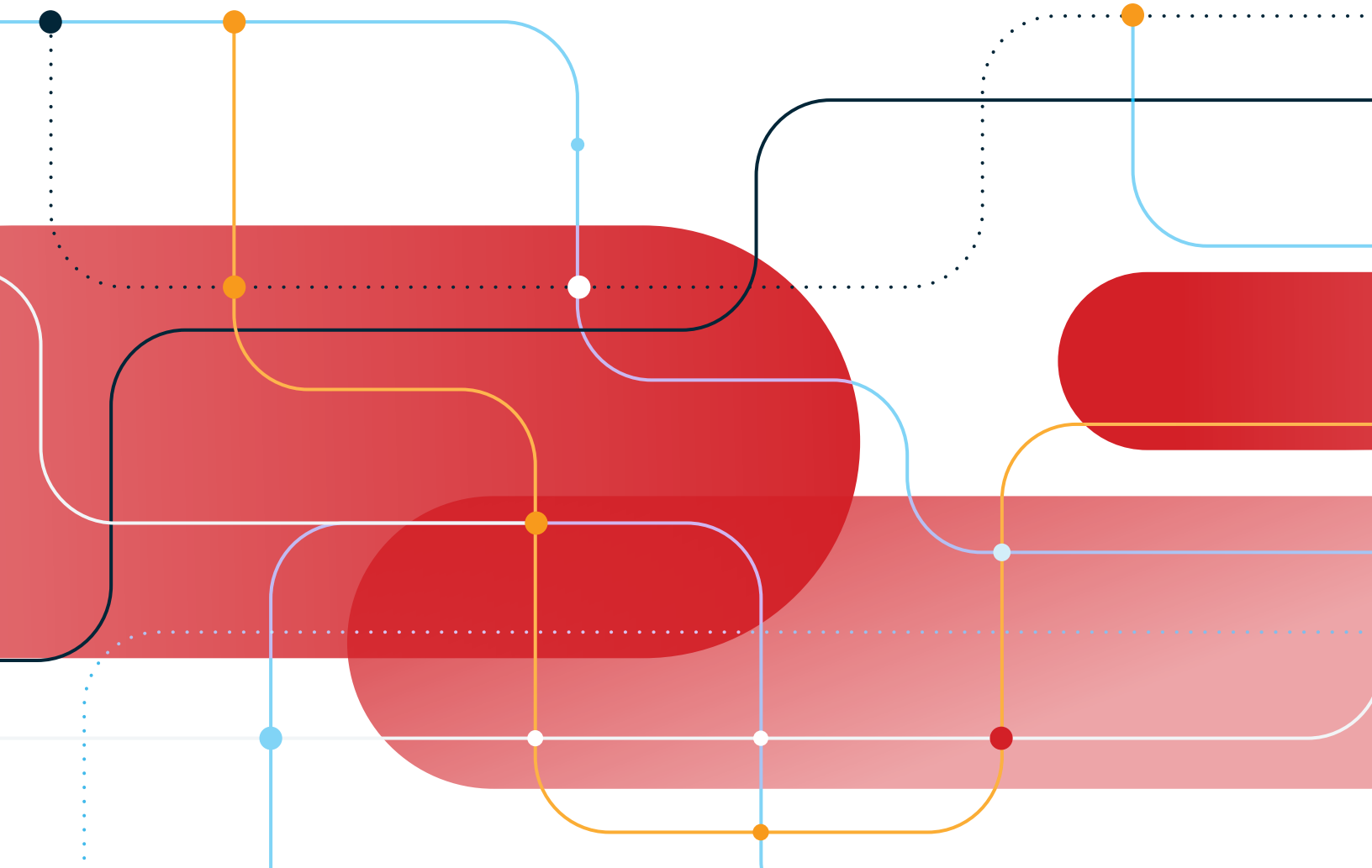
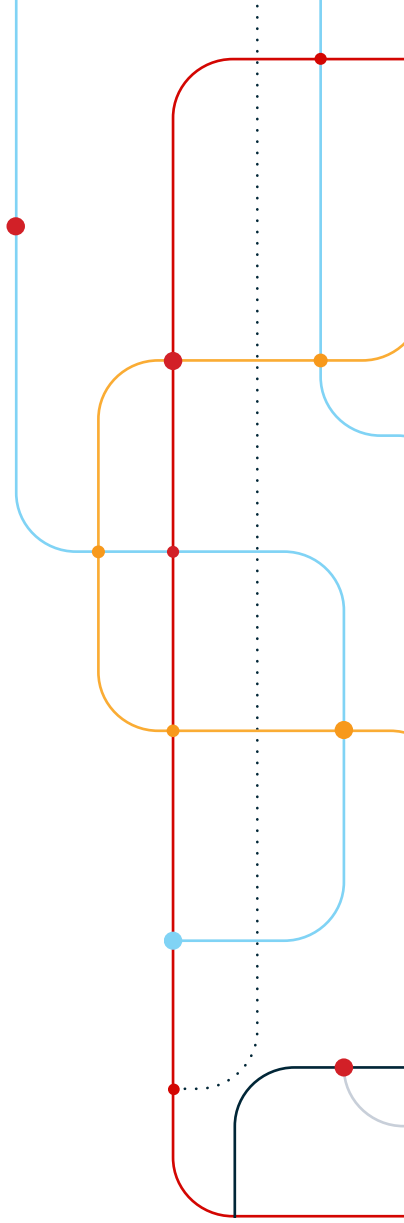
tecton

The Buyer's Guide to Evaluating ML Feature Stores & Feature Platforms



Contents



Introduction

Developing, extracting, managing, and productionizing feature data are some of the most challenging aspects of deploying machine learning models to production. Thankfully, solutions exist on the market simplify this process: standalone feature stores or end-to-end feature platforms. The question is, which one is right for your organization—and how should you evaluate particular products?

In this guide, we'll provide a comprehensive framework for understanding product capabilities to help you make an informed decision. We'll also share examples and tips for performing a data-driven vendor evaluation.

Debating whether to build a feature store in-house? Check out **our guide to the build vs. buy decision**.

Why Do You Need a Feature Store or Platform?

Before evaluating a specific product, it's important to have a clear understanding of the challenges you're looking to solve and the benefits you hope to achieve. These are the most common benefits that teams have achieved by adopting a feature store or platform:

Improve ML model accuracy. With a solution that consistently stores and serves features across your online and offline environments, your models will benefit from reduced training/serving skew. Furthermore, using standardized feature definitions that are write-once, use-everywhere reduces the risk of human error. Some feature solutions also offer monitoring and alerting on ML data and infrastructure.

Reduce the headcount required to build and manage ML models. Once it's set up and running, a feature platform can significantly reduce engineering overhead in several ways. First, almost all solutions promote feature sharing and reuse. Second, some solutions will automatically configure, orchestrate, and maintain data pipelines for you.

Unlock new, valuable real-time use cases using streaming or real-time data. Many organizations currently have the tools to use ML for offline analytical cases (like reporting or forecasting). But they often seek a third-party solution to enable real-time ML models that use streaming (e.g., Kafka or AWS Kinesis) or real-time data (like predicting live wait times, generating adaptive product recommendations, or detecting fraud). These models are very hard to productionize without specialized tools and infrastructure for serving features at low latency using real-time data.

Collaboration across ML teams and functions. When your organization has a larger and more distributed team, a feature solution can greatly improve collaboration. With capabilities like workspaces, feature sharing, CI/CD, and features-as-code, you can treat ML like software and empower data scientists and ML engineers to work together and iterate rapidly.

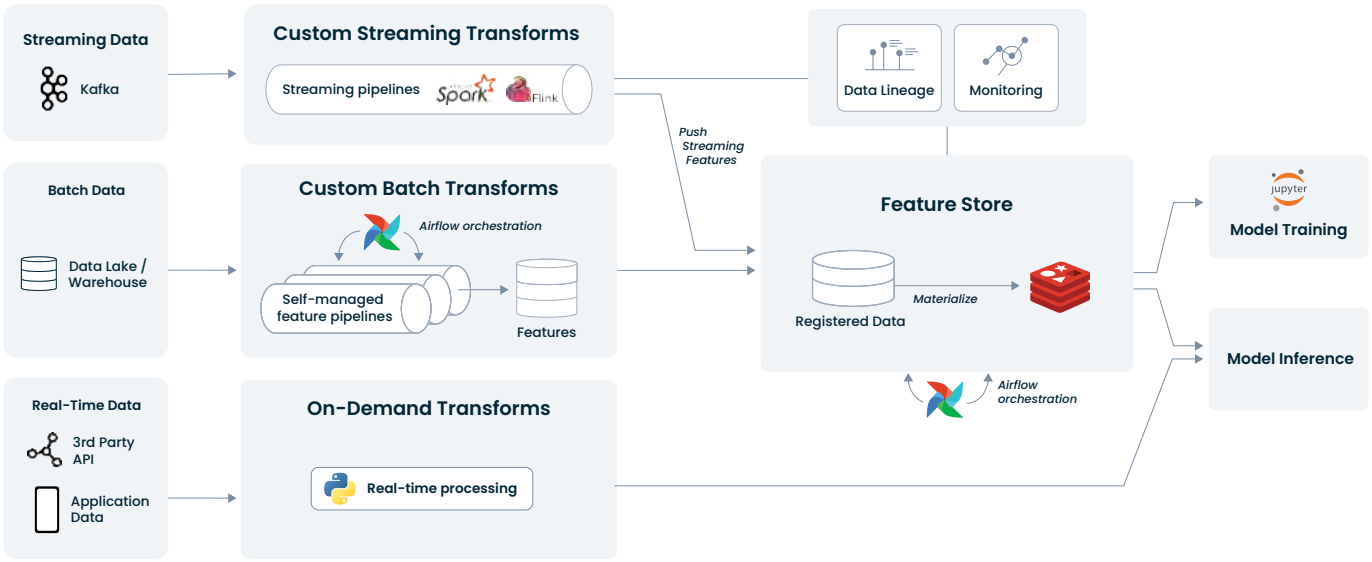
This all adds up to perhaps the biggest benefit that teams experience when they evaluate an ML feature solution: **reducing the time and effort to launch, improve, and maintain models in production.**

ML Feature Solution Categories

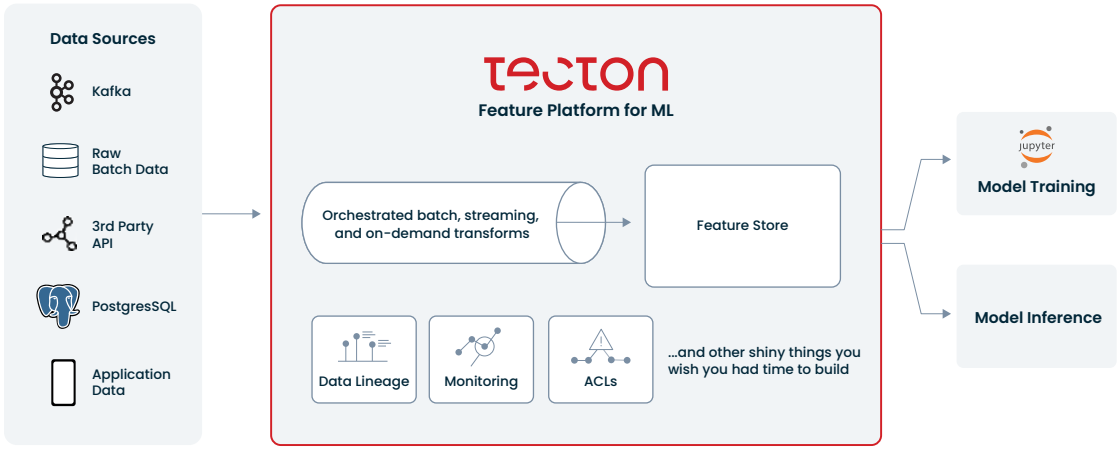
Until recently, a team developing and running multiple predictive applications in production would typically need to build some sort of feature management solution themselves. Today, the MLOps landscape is booming, and there isn't just one category of third-party options, but several.

Depending on how your data teams work today—and how they plan to work in the future—you might find that the best fit for your needs may be a feature store, a feature platform, or an integrated MLOps platform that includes a feature store. If you're already familiar with the differences and common tradeoffs between these three categories, skip ahead to the **Evaluation Process** section.

Feature Store Architecture



Feature Platform Architecture



ML Platform With Feature Store Architecture

Feature store

A feature store is, simply put, a central system to store and serve feature data that has been processed by external, self-managed pipelines. The feature store ensures consistency between offline and online data, and provides central APIs to generate accurate training datasets and serve data online for real-time inference.

Since feature stores are the most lightweight of the three categories, they tend to be the easiest to integrate into an existing MLOps stack. There are open-source, self-managed feature stores that are completely free to try—Feast is one example.

However, the downside is that most feature stores don't enable you to automatically transform and materialize features from your data sources, meaning you still need to manage data pipelines to transform raw data into feature values. This can be especially challenging if you are trying to compute features from real-time and streaming data sources, then turn around and serve them to models with low-latency inference requirements.

Feature platform

Emerging out of the need for a more comprehensive solution than a feature store, a feature platform manages the complete lifecycle of ML features, from design to production. A feature platform extends a feature store to automate the data pipelines that process data from batch, real-time, and streaming data sources and for model serving requirements. And it typically includes other enterprise features like CI/CD readiness, enterprise security and permissions management, separate workspaces, and monitoring. Tecton is an example of a feature platform.

Compared to stand-alone feature stores, feature platforms cover a more exhaustive list of capabilities and are typically more powerful. They are a better choice if you're looking to automate the complete lifecycle of features, including feature pipelines. Many feature platforms also allow you to ingest data directly from external pipelines, thus providing more flexibility to integrate with existing data infrastructure.

Feature platforms greatly accelerate the time to delivery of new models. Compared to end-to-end ML platforms, feature platforms tend to be more specialized since they handle just one part of the ML lifecycle. A feature platform is designed to be modular and work with your existing ML stack, so it can be adopted incrementally, adapting well to existing systems, tools, and frameworks. (You can also see a side-by-side comparison of Tecton and Feast here.)

MLOps platform

As the MLOps tools market has exploded over the past few years, we have seen a rise of platforms designed to provide a holistic solution for ML needs. They support the entire ML lifecycle end-to-end with tools for model training, serving, and monitoring—and some of them also include a feature store. **Amazon SageMaker** is an example of an MLOps platform that has a fully managed feature store.

For a new ML team, an integrated MLOps platform can be an attractive option. It handles the work of choosing different MLOps products and building a well-integrated stack. On the other hand, if you already have components of an ML stack in place, you'll need to transition it over to the platform, which can be very time-consuming (and switching out of the platform later can be equally difficult—i.e., this may not be the best option if you want to avoid vendor lock-in). And because ML features are just one component of an MLOps platform, the solution may not have the specialized capabilities of a feature platform.

Evaluation process

Evaluating an ML feature solution is a multi-step process that takes time and careful consideration. To have the most success implementing the framework above, we recommend the following:

Involve your team. Before starting your evaluation, we recommend using the framework as a starting point to interview colleagues involved in your ML efforts—data scientists, machine learning engineers, data engineers, DevOps, security/compliance, product managers, execs, and more. This exercise gives you valuable information about your organization's priorities and it's also a great way to generate buy-in.

Define your criteria upfront. What's a much-needed capability, and what can you do without? With the information you've gathered from interviews, create a scoring system for different capabilities that lets you measure how well vendors fit your needs. Develop your criteria with an eye toward the next few years to ensure your solution doesn't become obsolete by the time you're done integrating it.

Feature Platforms Compared	tecton	Vendor 2	Vendor 3
Vendor Quality	Small but most mature	Good	Good
Feature Tagging	✓	◑	◑
Search	✓	◑	◑
Version Control	✓	✗	✗
Compatible Formats	✓	✓	✓
Online + Offline Store	✓	✓	✓
RBAC	◑	◑	◑
Databricks Integration	✓	◑	✓
Spark Integration	✓	✓	✓
AWS Integration	✓	✓	✗
Automated Batch, Streaming, and Real-Time Transformations	✓	✗	✗
Technical Scores Max Score: 430	388 (90.2% of max)	274 (63.7% of max)	270 (62.8% of max)

Key: ✓ Fully Covered ◑ Partially Covered ✗ Missing

Conduct a POC that delivers quantifiable results. A POC with a vendor is a must for evaluating any new software. But make sure to design the POC so that you end up with objective benchmark data rather than just subjective personal impressions. One way to do this is to duplicate an existing use case in the new solution to A/B test implementation with the vendor vs. your current implementation.

Achieve a quick win. After you've made a decision, it can be useful to pilot your new tool on only one or two key use cases. Any change is difficult, and your team will need some time to adapt to new ways of doing things. With a contained use case, you can get a quick win and start to prove out the value of your solution right away.

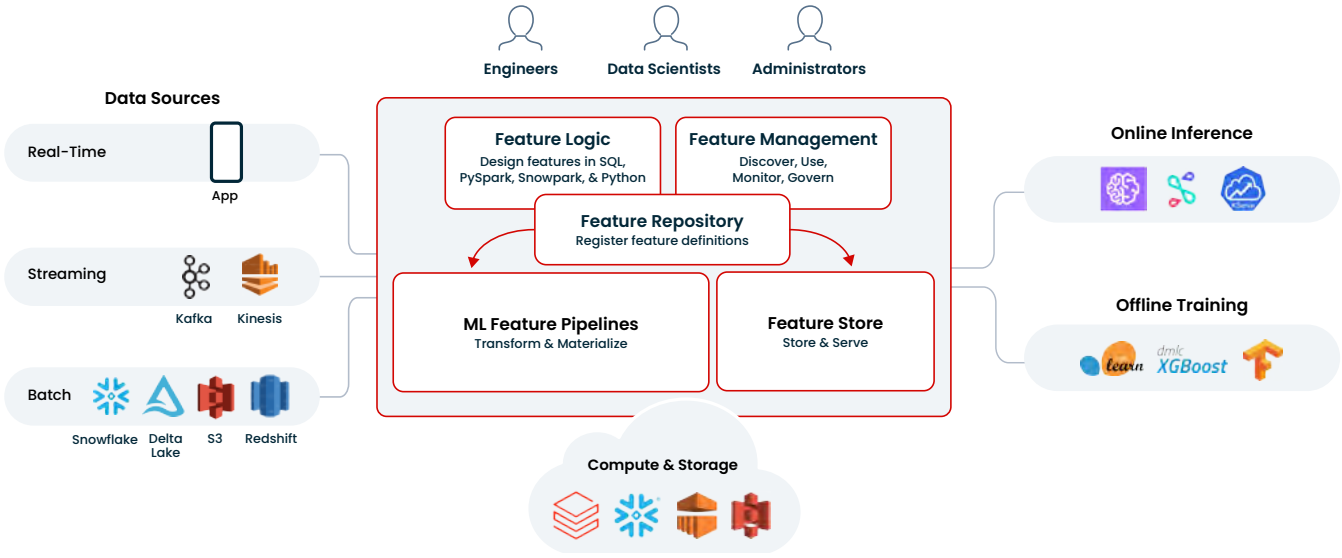
Involve your team (again). Just as you did at the beginning of your evaluation, when you're ready to implement your feature solution, make sure that you're bringing all the appropriate stakeholders on board. Internal training, demos, and documentation are essential to align your ML team(s) on the objectives and roll-out of your feature solution.

Evaluation framework

Choosing a feature store, like deciding to use any new piece of software, is about making tradeoffs based on your organization's needs. But because feature stores, feature platforms, and MLOps platforms are new categories of tools, the tradeoffs aren't always well-established—or even clear to begin with.

To help you determine the best solution for you, we put together a framework that identifies 1) the core capabilities of an ML feature solution and 2) enterprise considerations like support, security, and the deployment model.

We've included notes to provide you with extra context and the common questions we hear from real teams conducting their own evaluations. To help you decide between the must-haves and the nice-to-haves, we've also included questions at the end of each section that can be used to prompt internal discussion.



Core capabilities

	What it is	Why it matters	Common questions
Data Sources	How the solution supports third-party data sources and APIs.	Makes it possible to derive features from a number of different streaming, batch, and real-time (application) data sources.	• What batch data sources are supported? • What streaming data sources are supported? • Can it ingest real-time application data?
Feature logic + repository	The logical definitions and metadata needed to compute features on an ongoing basis.	Defining features as code helps standardize DevOps best practices such as version control, CI/CD pipeline integration, and code review across the organization.	• Are features defined as code? • How easy is it to create/update feature definitions?
Feature management	How teams discover, use, and govern features.	Features are the heart of ML models. A good ML feature solution should provide tools to facilitate feature sharing, reuse, and development.	• Is there a CLI? • Is there a web UI? • Are features reusable and versioned? • Can you search and explore features? • Can you separate out workspaces/ pre-release environments? • Is feature lineage supported?
ML feature pipelines	How (and if) the solution orchestrates pipelines to compute and materialize features, using automated transformations on raw data.	When a solution handles orchestration for you, you avoid having to maintain complex data pipelines for ML. This can be especially beneficial when using real-time and streaming data.	• Can you automate batch, streaming, and real-time pipelines? • How is online/offline parity ensured? • How is data backfilled? • Are on-demand transformations supported?
Offline feature store → Offline training	How small, more current batches of features with low-latency requirements are served to models in production for real-time inference.	Low-latency online feature serving makes it possible to power your applications with real-time ML predictions.	• Can you query online features with a single line of application code? • How fast is feature retrieval? • What is the API? • How many QPS is supported? • What are the SLAs? • How is feature freshness handled? • What third-party integrations are supported?

	What it is	Why it matters	Common questions
Compute and storage	How the solution stores and processes features under the hood (e.g. data lake/data warehouse for storage, Spark for processing)	You may have preferences based on your existing data stack and performance needs. Flexibility and pluggability are also important to future-proof the solution.	• What technologies are used for storage and processing? • What optimizations are supported to reduce costs? • Can you configure the feature store architecture?

Ask your team:

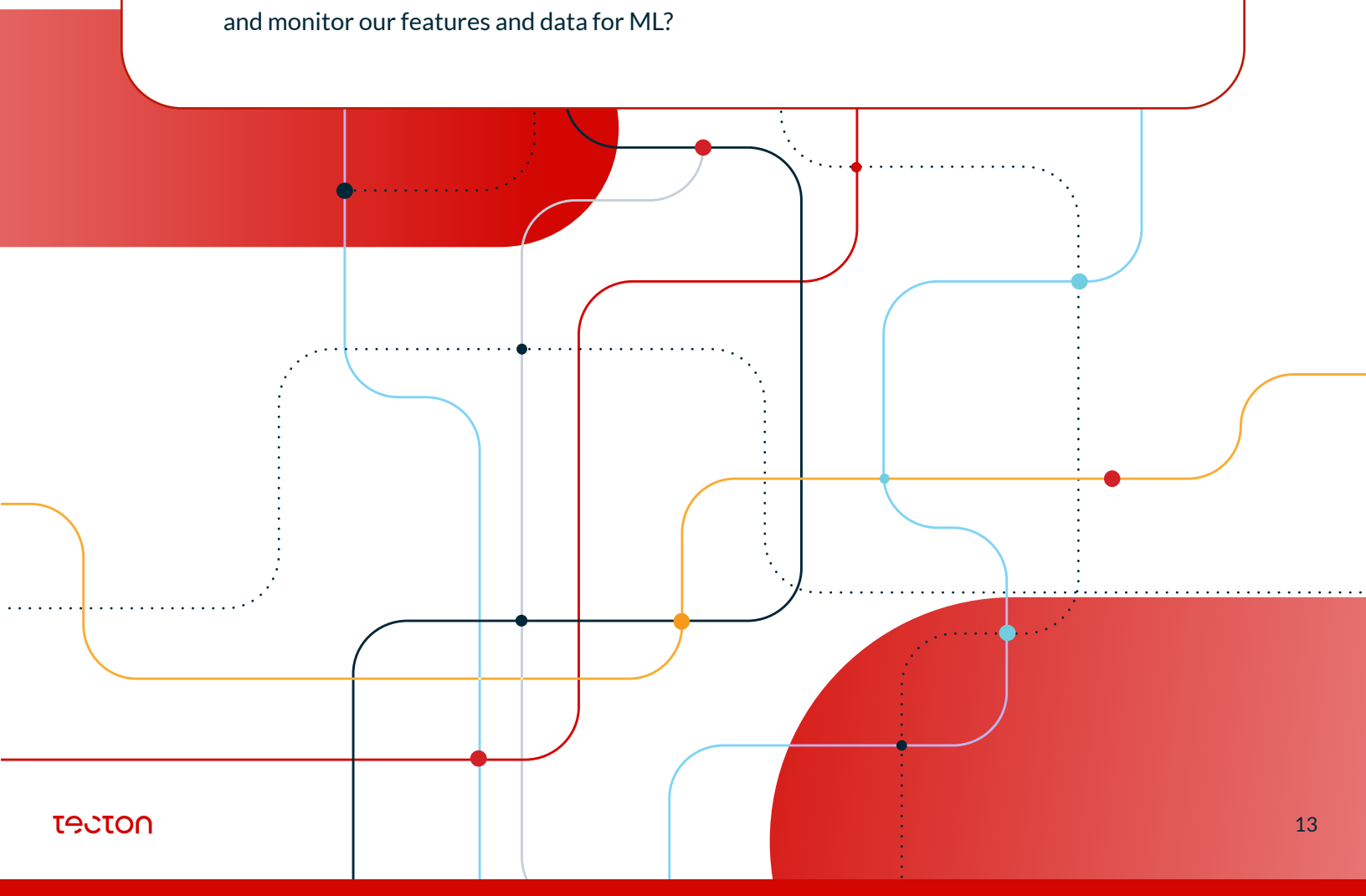
- What type of data (batch, streaming, and real-time) do we currently use or plan to use for ML use cases in the future?
- What are the primary ML use cases we are building toward today—and what use cases do we anticipate building towards? Do those use cases require real-time inference?
- At what scale do we plan to deploy our ML use cases? (Number of users, number of models, QPS.)
- How many data scientists or data engineers do we have, and how are features shared and reused today, if at all?
- Can a feature platform answer all of our team's technical requirements, including latency and volume, especially when it comes to more complex real-time use cases?
- What are the cost implications in terms of employee count, internal resources, number of hours, when it comes to building and maintaining additional ML pipelines with in-house vs. managed solutions?

Enterprise considerations

	What it is	Why it matters	Common questions
Delivery model	How the solution is deployed and maintained (e.g. fully-managed SaaS vs. self-managed open source)	This will determine how well the solution fits with your current stack and how much of the maintenance burden falls to your team.	• Is it self-managed or fully managed? • Do they offer a hybrid deployment? • What clouds are supported?
Support	The support, services, and guarantees a vendor provides to ensure the reliability of their solution	Feature solutions serving real-time production ML use cases are often considered to be mission-critical for enterprises.	• What are the SLAs? • Will you be able to talk to a human to get help with your production model pipeline?
Pricing model	How the vendor charges for the solution	Depending on your ML use cases, different pricing models can yield vastly different total costs.	• Is pricing done by node, by user, or by consumption? • Does pricing scale linearly with usage? • Does the product optimize the costs of the underlying processing and storage?
Security + access controls	The vendor's certifications, security practices, and in-product controls for protecting your integration	Feature data is often sensitive, proprietary customer and user data that is essential to protect.	• What auditing capabilities are offered? • How does role-based data access work? • Is SSO/OAuth supported? • Are they SOC 2 compliant? • How is data encrypted?
Monitoring + alerting	How feature data and infrastructure is tracked to monitor data distributions across training and prediction pipelines to generate alerts if data diverges	Since your ML feature solution likely represents mission-critical infrastructure, you need to be immediately notified of issues.	• Is infrastructure monitored? • What are the options for alerting? • What metrics and visualizations are offered? • Is there tracking for data quality or model quality issues?

Ask your team:

- In what type of environment are we looking to deploy our solution, and how might this evolve over time?
- Are we willing and able to maintain a self-managed solution that meets our requirements for level of service and uptime?
- How do we anticipate the number of accounts/number of nodes/amount of consumption changing over time at our organization?
- Do we prefer to pay for exactly what we use each month or have more predictable spend for budgeting purposes?
- What SLAs and support do we require for our use cases in production?
- Are there regulatory requirements in our industry that are relevant to how we handle and monitor our features and data for ML?



Putting It All Together: A Real-World Example

To give you an idea of how another company put their evaluation process into practice, here's how HelloFresh evaluated feature solutions for ML:

1. HelloFresh didn't want to replace a large number of ML tools they were already happy with, so they ruled out ML platforms from the start and opted to look for a standalone feature store/platform that would be more specialized.

2. They brought together their data science teams and held brainstorming sessions to quickly identify the top topics of concern.

3. Then, they translated common topics into a list of evaluation questions such as "What data formats can be stored as features?" Each question was assigned a weighted priority level from 1-5.

4. To narrow down their top choices, HelloFresh reviewed vendor websites and whitepapers. They chose to demo with the vendors whose technical capabilities scored highest on their objective criteria.

5. For their POC, HelloFresh picked one of their machine learning models and uploaded its features to different vendor solutions so they could test different scenarios.

6. The team then presented their quantitative results to data leadership, explaining how the platforms scored on their criteria and the associated tradeoffs.

Key Takeaways

Evaluating feature solutions for ML can be a challenge. First, you need to consider not one, but three categories—**feature stores, feature platforms, and ML platforms**—to narrow down the right fit. Within these categories, you'll find solutions with a variety of different capabilities, relating both to how they work from a technical standpoint, and how they fit into your organization with regard to security, deployment, and support.

Every organization is different, and there's no right answer to how various ML feature solution capabilities should rank on your priority list. However, we hope that the framework in this guide, along with the accompanying questions, provides a basic foundation that you can adapt and tweak until it's right for your data and ML team.

Have any suggestions? We'd love to hear them. And if you have additional questions, we're here to help (as we've done with many other ML teams in their journey to putting models into production faster). Feel free to contact us, or if you'd like to see Tecton in action, you can watch a demo here.