Stream Everything

Dean Wampler, Ph.D.
dean@lightbend.com

HPC on Wall Street

Streaming in Finance

Why streaming matters for Finance.

- Algorithmic Trading
- Continuous Risk Management
- Continuous Monitoring
- Continous Customer Engagement
- Live Data Marts

Mentioned by Philip Filleul.

Algorithmic Trading

- For non-HFT applications, more options are available for:
  - Machine learning
  - Complex event processing (CEP)

HFT requires customized, very low-level implementations to satisfy the latency requirements. If you have less-stringent latency requirements, you can exploit a rich ecosystem of tools. I'll describe some example open-source software tools shortly.

Continuous Monitoring

- Risk Management
- Fraud Detection
- Tradeflow
- Compliance

I’ve tweaked Phil’s list a bit here.

Risk Management

- Update models frequently
- Enforce model constraints

An emerging industry pattern is the need to train models on “fresh” data indicating evolving threats. This doesn’t necessarily need to be “real-time”, but usually much more frequently than daily. However, scoring for threat signals needs to be fast.

Fraud Detection

- Similar challenges to Risk Management
- Update models frequently
- Score data quickly against the model

Similar to risk management, an emerging pattern is the need to train models on “fresh” data indicating evolving threats. This doesn’t necessarily need to be “real-time”, but usually much more frequently than daily. However, scoring for threat signals needs to be fast.

Tradeflow

- Reliable capture of high-volume streams
- Fan out to downstream consumers

Many traditional data stores aren’t really optimized for reliable, scalable capturing of high-volume streams of data. Once captured, downstream consumers need fast access that scales easily with the number of consumers.

Compliance

- Reliable capture of high-volume streams
- Fan out to downstream consumers
  - Real-time enforcement
  - Ex post facto auditing

Continuous Customer Engagement

- Near instant car loan approvals.
- Mine historical data, join with real-time data.

Some specific

Data Marts

- Ideally the same system that captures and processes streams provides access for “offline” analytics.

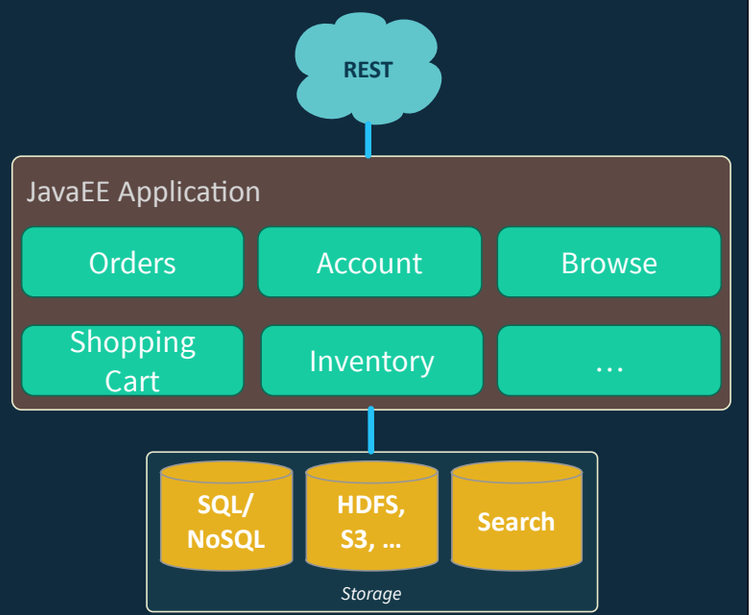
This could also include streaming data out to consumers, such as customers.

How do we implement these capabilities?

Building Services

Monoliths

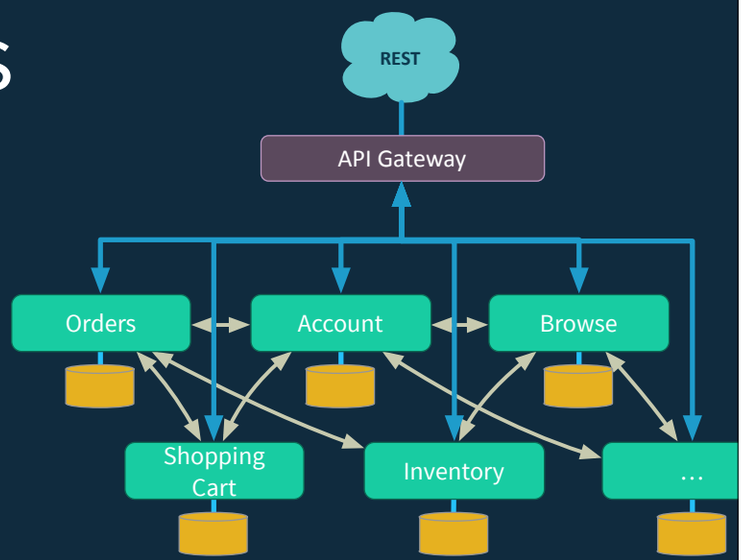
- Tangled responsibilities.
- Hard to evolve.



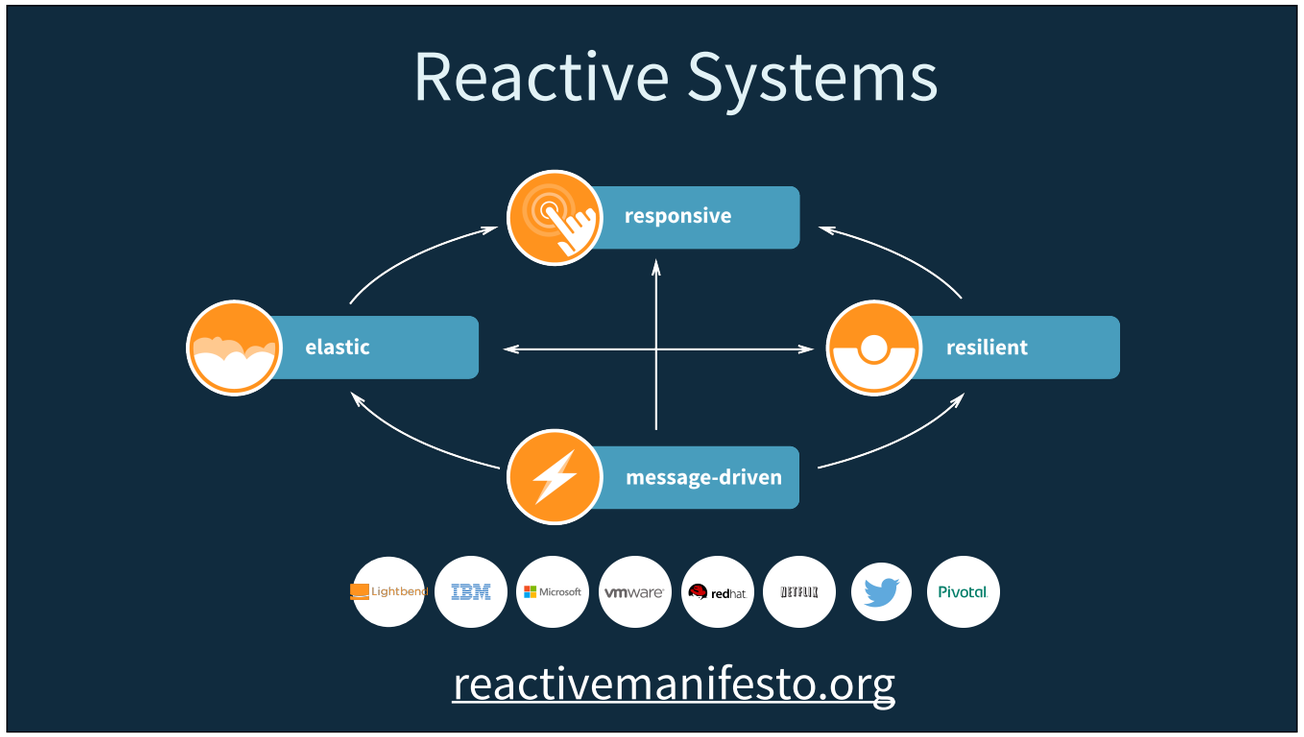
A classic, JEE approach for “macroscopic” services (monoliths). App Containers are too heavyweight to run one per microservice. So, you have a lot of concerns and dependencies in one place. That means it’s difficult to evolve those features separately, leading to “big-bang”, infrequent deployments.

Microservices

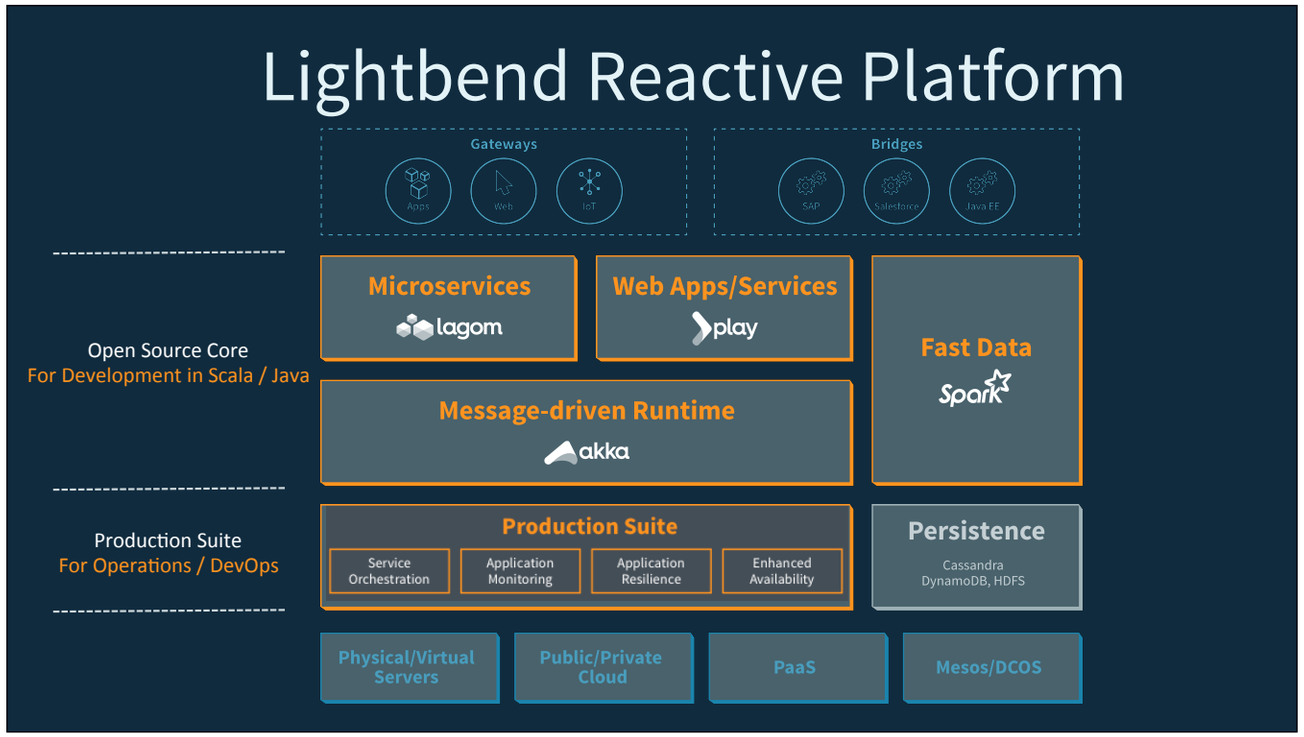
- Single responsibilities.
- Easy to evolve.



Microservices have single responsibilities, so they are easier to evolve separately, scale up/down, isolate, etc. They must communicate with each other to get services they need, which is best done through asynchronous messaging.



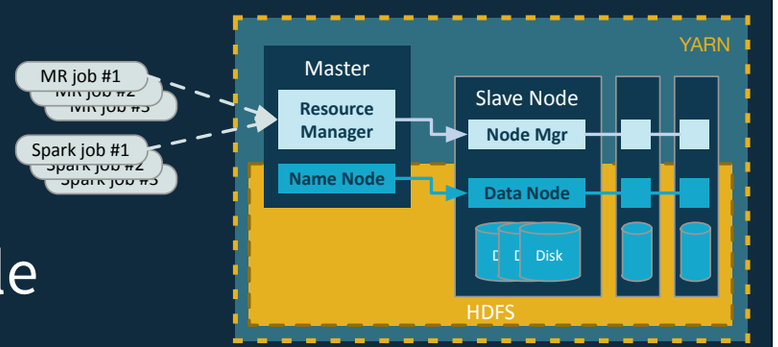
But because stream systems run for a long time (days to months to even years), they have the same “reactive” requirements as microservices, elastic in response to demand, responsive even when degradation has occurred, resilient against failures, and message driven to support all of they above. The reactive manifesto, which defines these traits, has wide adoption in industry.



The Lightbend Reactive Platform. Lagom is an “opinionated” API that promotes best practices for microservice design and accelerated developer tooling. Play is a full-stack web framework. Akka is an Actor-based (message based) runtime and suite of libraries. RP current includes Spark support, but we’ll discuss our expanded Fast Data platform below. Production suite provides essential, RP-focused production tools.

What about
Data?

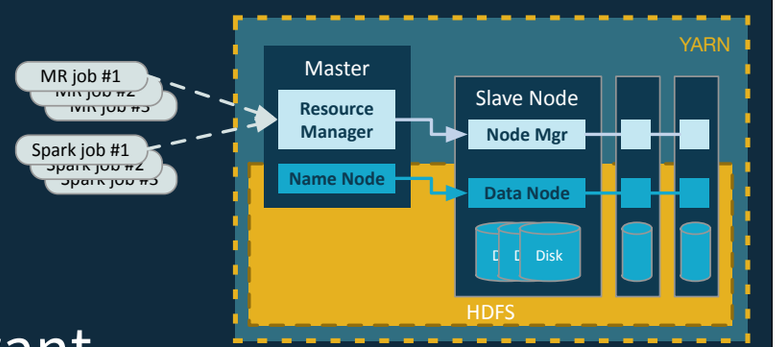
Hadoop



- Low cost, scalable batch analytics.
- Data warehouse replacement

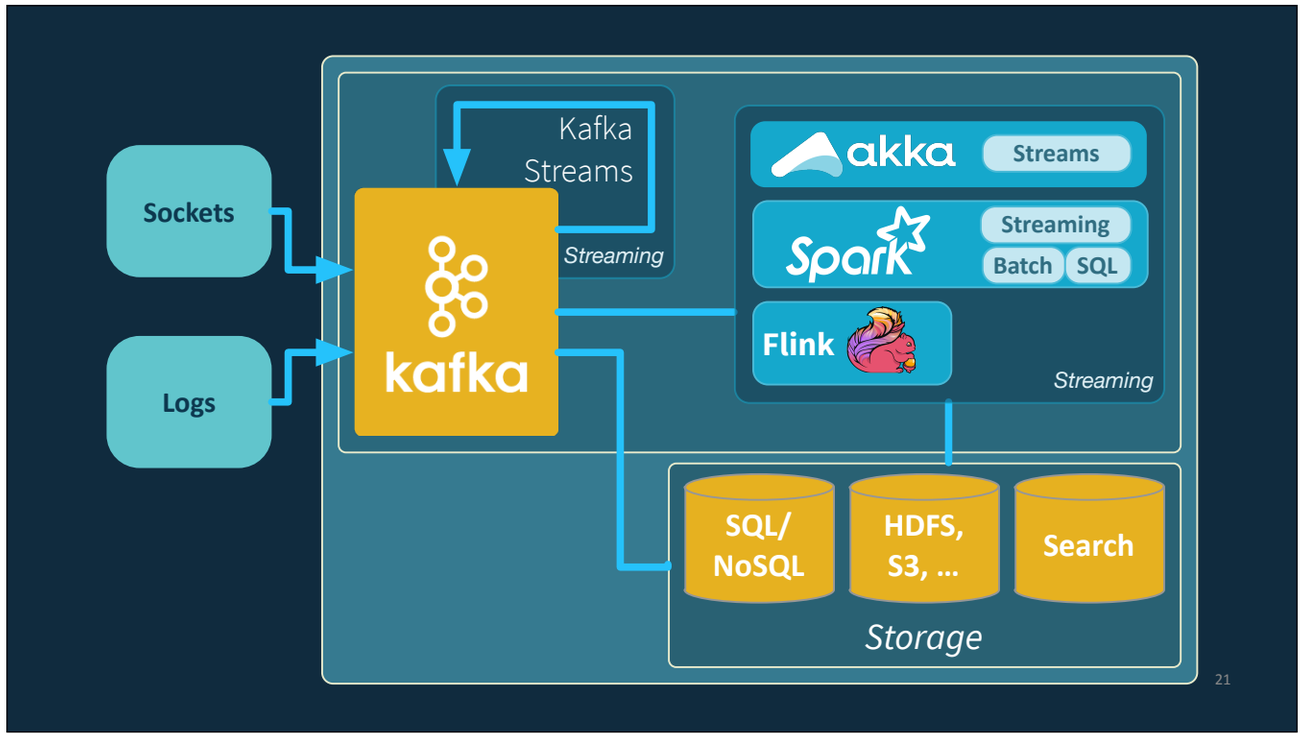
Hadoop emerged as a way of collecting, managing and analyzing massive data sets. It emphasized batch-mode processing. It’s become a data warehouse replacement (better scalability at dramatically lower cost).

Hadoop



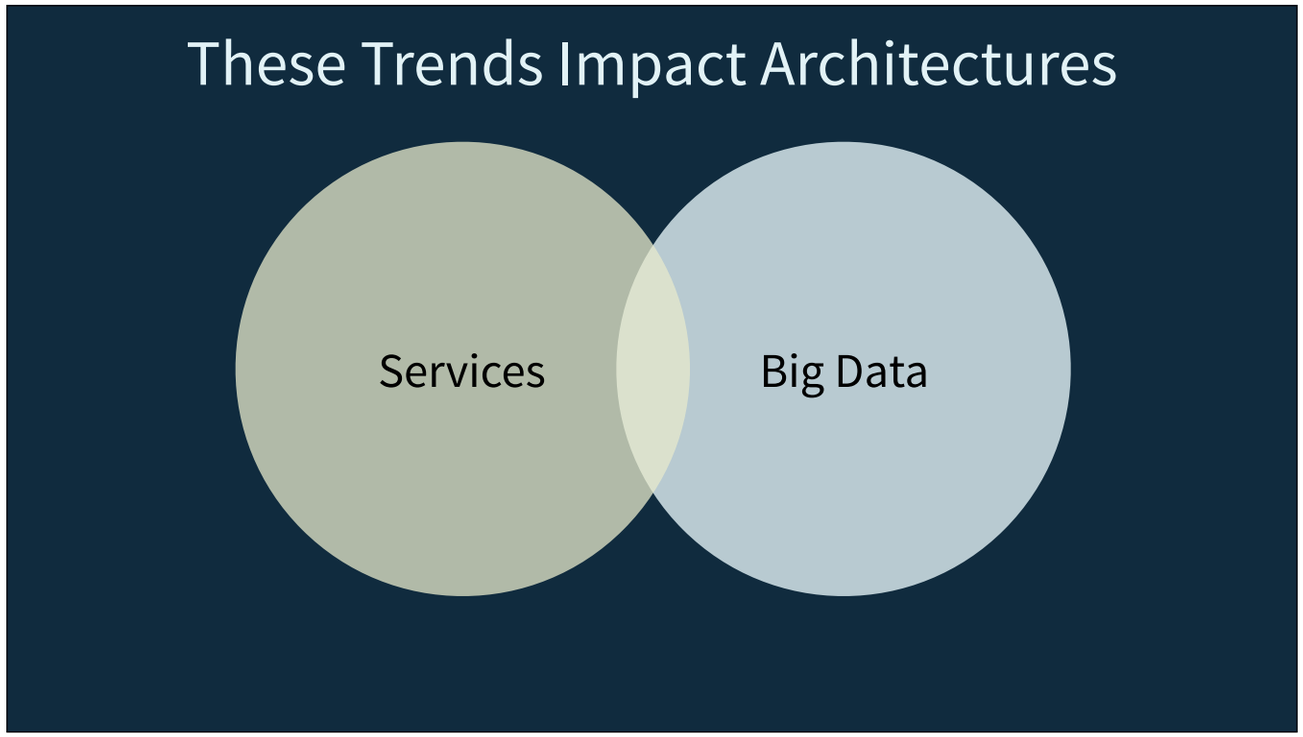
- … but what if I want answers soon?

➜stream processing

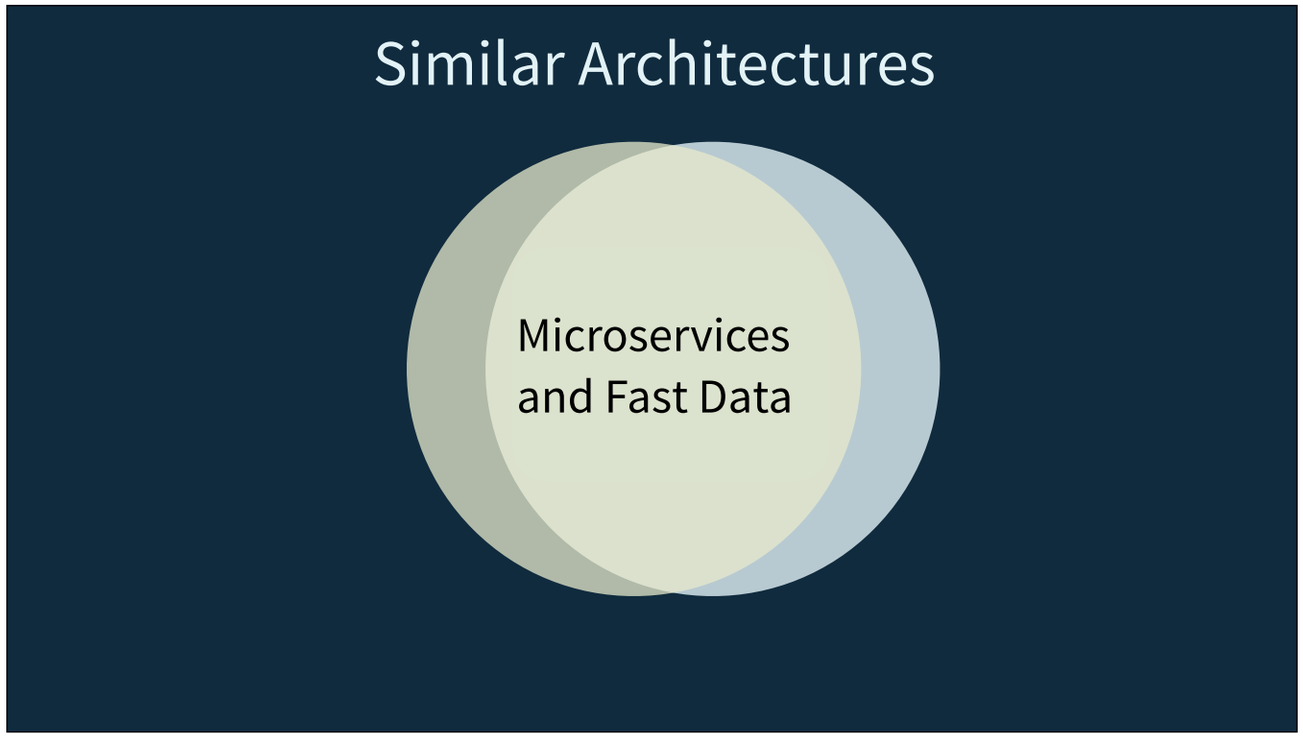


That means we need a backplane designed from streaming. We'll see that Kafka is ideal for that purpose. We also need a variety of stream processing engines, because no single engine meets all needs. Finally, persistence is required.
We'll explore the elements in this diagram shortly, but first, let's look at a little microservice history.

Fast Data and Microservices; Are they Converging?



Until recently, people building “canonical” services for general-purpose IT apps have focused on high availability, scalability, resilience, etc. (i.e., the Reactive principles), recently moving to microservices to do this better.
The Big Data world has focused on data storage and cluster scalability, with less need to worry about the Reactive principles.
Of course these was some overlap, but they were different spheres…



IT apps have been evolving towards finer-grained microservices and offline Big Data environments have been evolving towards online, stream processing or “Fast Data” environments. I’m going to argue that the architectures for these two spheres are converging towards the middle.The Big Data world has focused on data storage and cluster scalability, with less need to worry about the Reactive principles.
Of course these was some overlap, but they were different spheres…

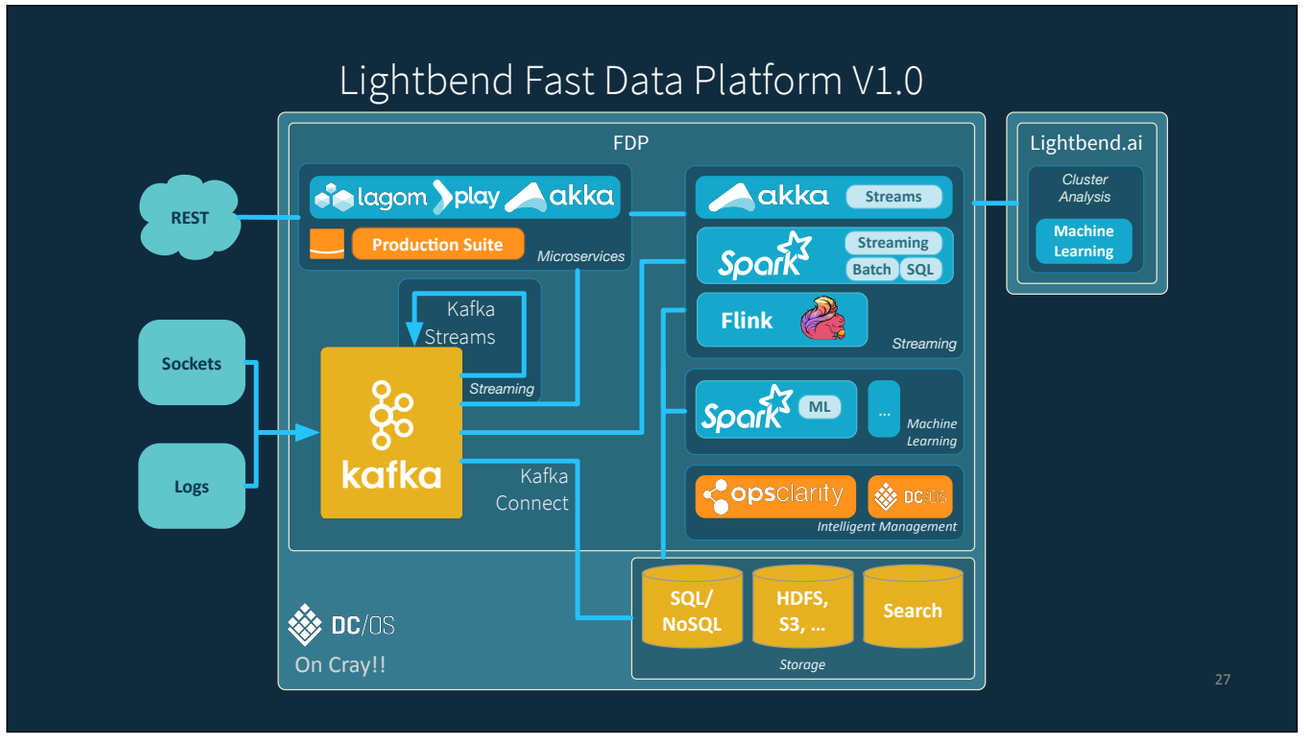
Lightbend
Fast Data Platform

Our vision is to provide an integrated platform for the spectrum of microservice and fast data systems people need to build today.

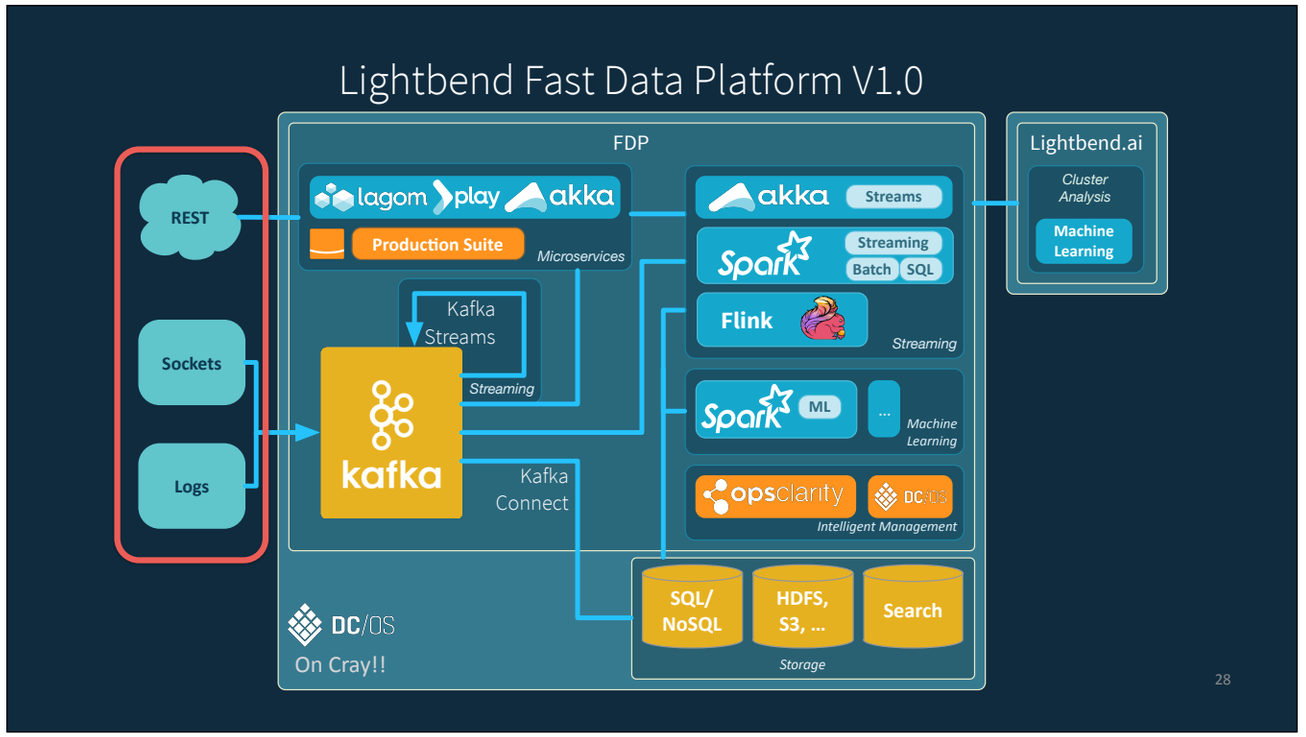
Value Three Ways

1. Accelerated on-ramp
2. Expert guidance to build your apps
3. Machine Learning to manage your cluster

There are three major ways in which FDP provides value. We'll explore each of these next.

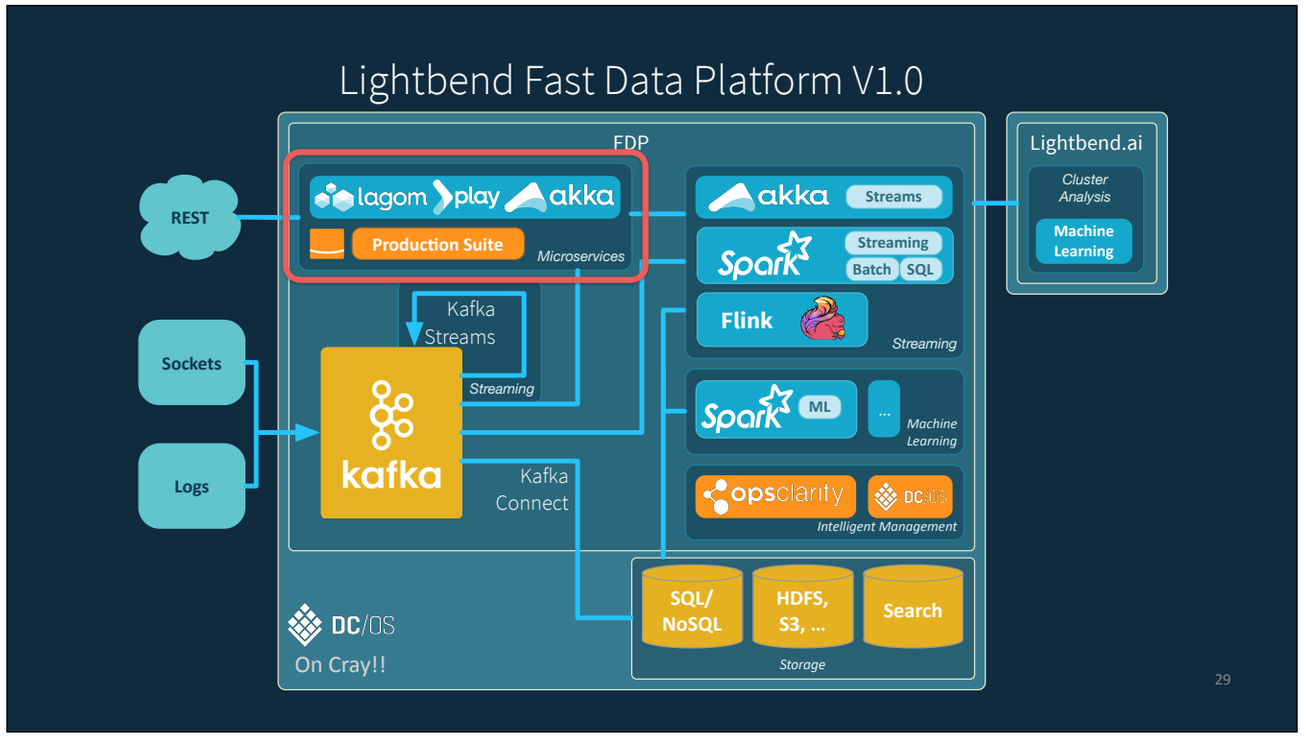


The big picture of FDP. Note that it includes the subset of streaming elements we saw before. Let’s explore the details…

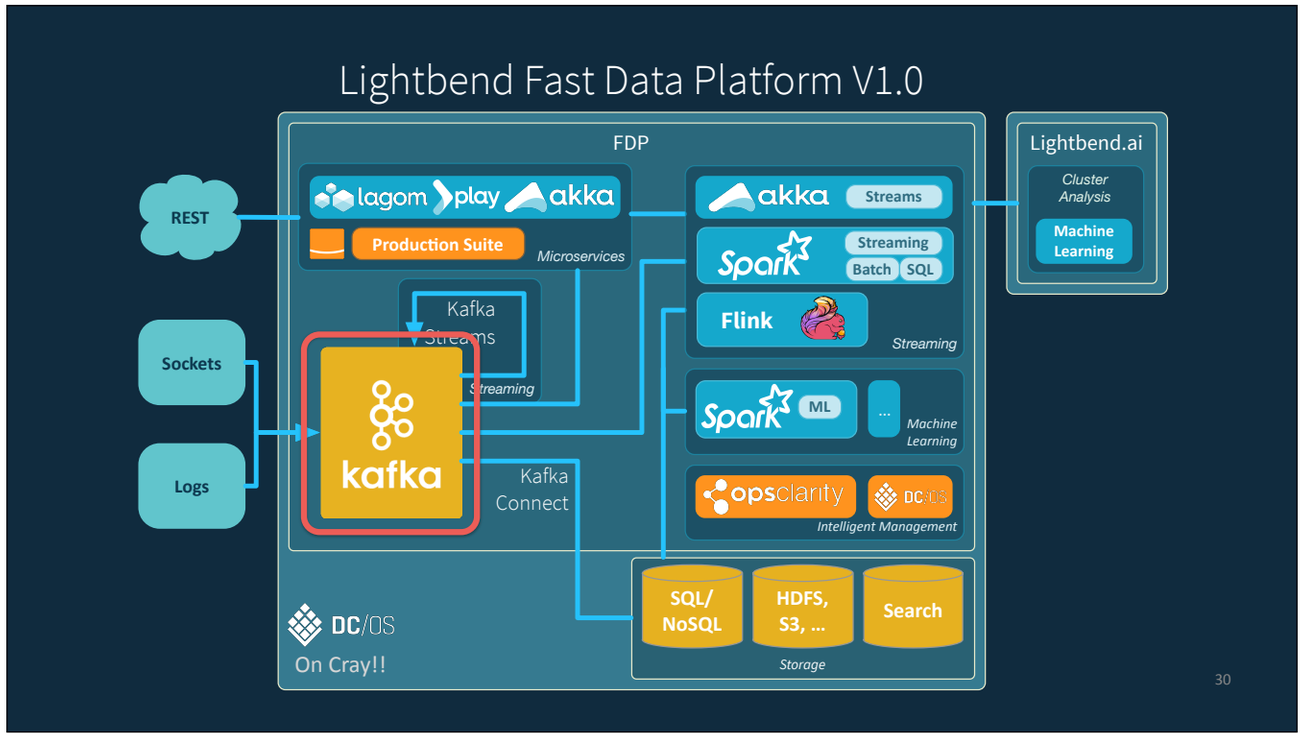


Input could be REST/HTTP through Lagom/Play-based microservices, but mostly you don't want this overhead for large-volume data streams. However, it's great for "control" messages, consoles, etc.

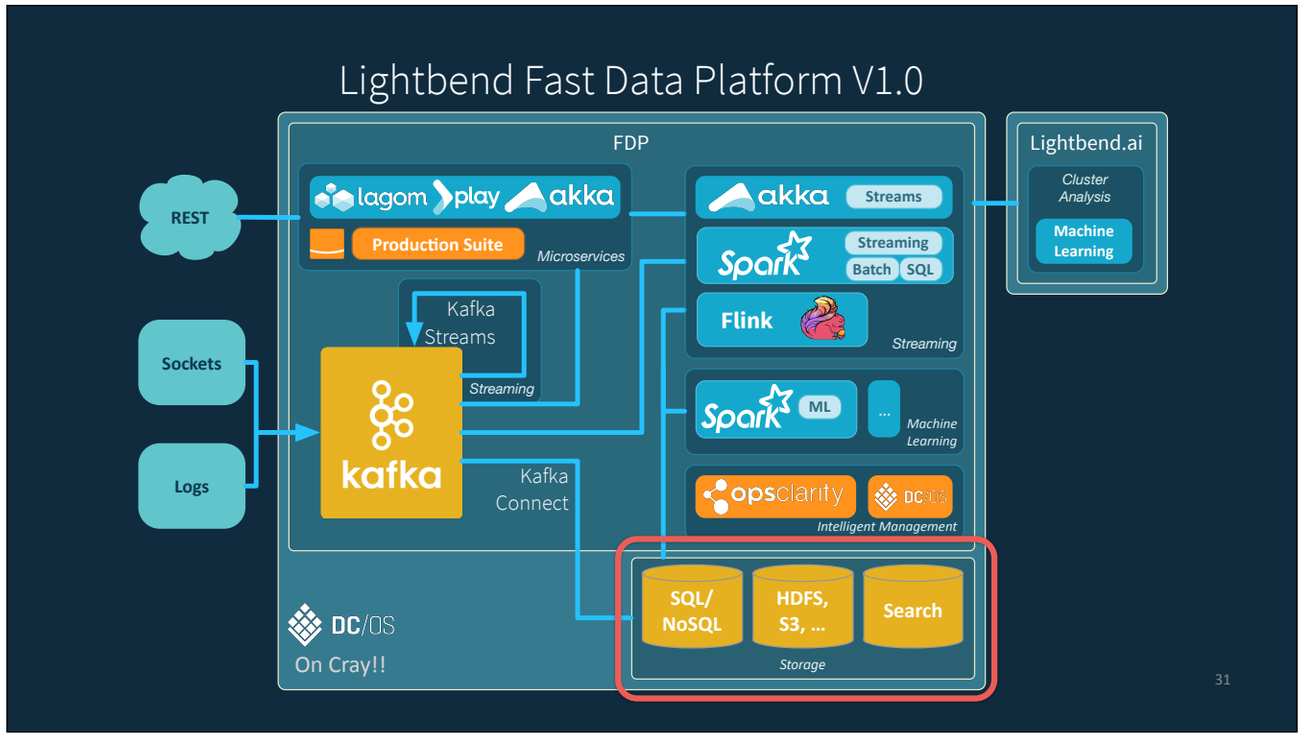
Otherwise, data streams are via sockets (e.g., Twitter fire hose, IoT device telemetry) and log ingestion.



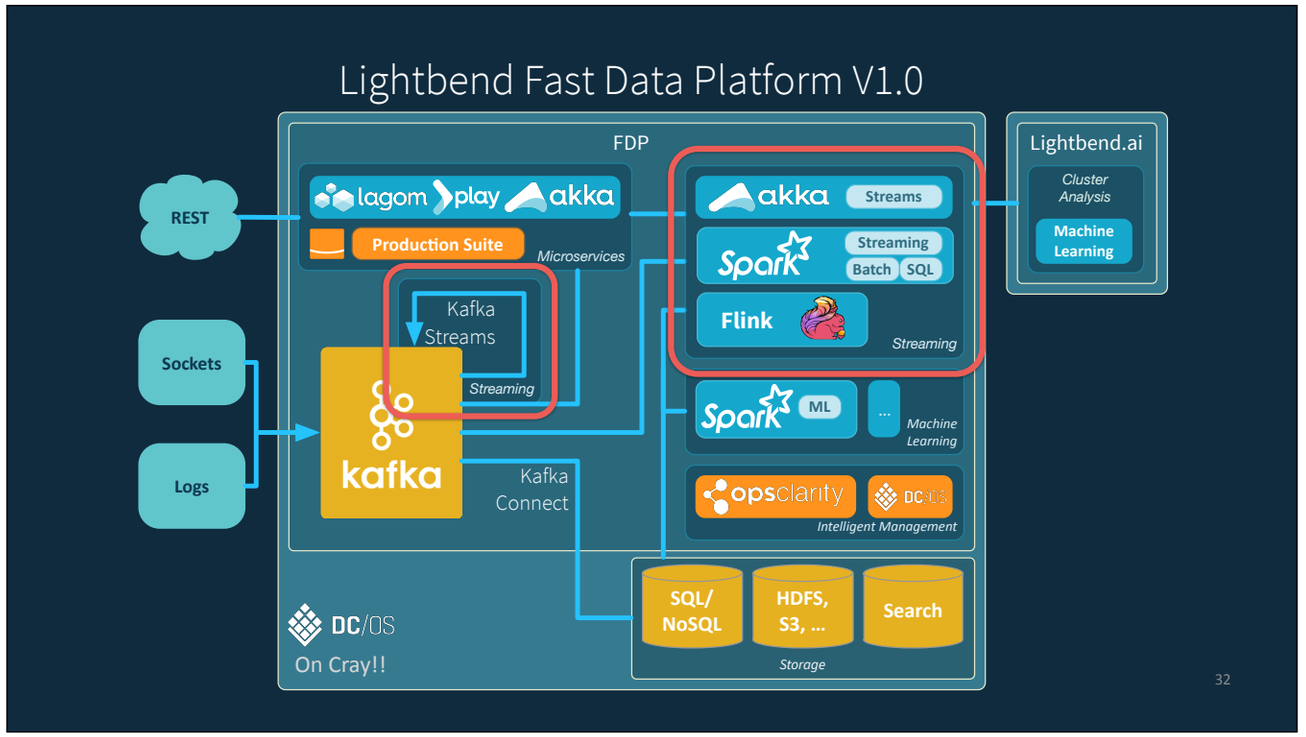
Any real environment needs some custom microservices to provide all the other capabilities a real environment requires. Use RP for this, which is integrated with the rest of FDP and supported by Lightbend…



Kafka is the backplane, providing integration, short-term durability, pub-sub model, etc.



Storage options (provided by DC/OS) include various SQL and NoSQL databases, distributed file systems like HDFS (Hadoop Distributed File System), object stores like S3, and search engines like Elasticsearch.



Now let’s talk about the streaming engines…

Streaming Tradeoffs (1/3)

- Low latency? How low?
- High volume? How high?

Several considerations when choosing tools.
Some tasks require a few microseconds are less, while others can tolerate more latency, especially if it allows the job to do more sophisticated or expensive things, like train machine learning models iteratively, write to databases, etc.
At high volumes, you might have to pick a very scalable tool with amortized excellent performance per event, but not when processing low volumes (e.g., due to the infrastructure it uses to support high volumes). Alternatively, a tool with excellent per-event performance might not scale well.

Streaming Tradeoffs (2/3)

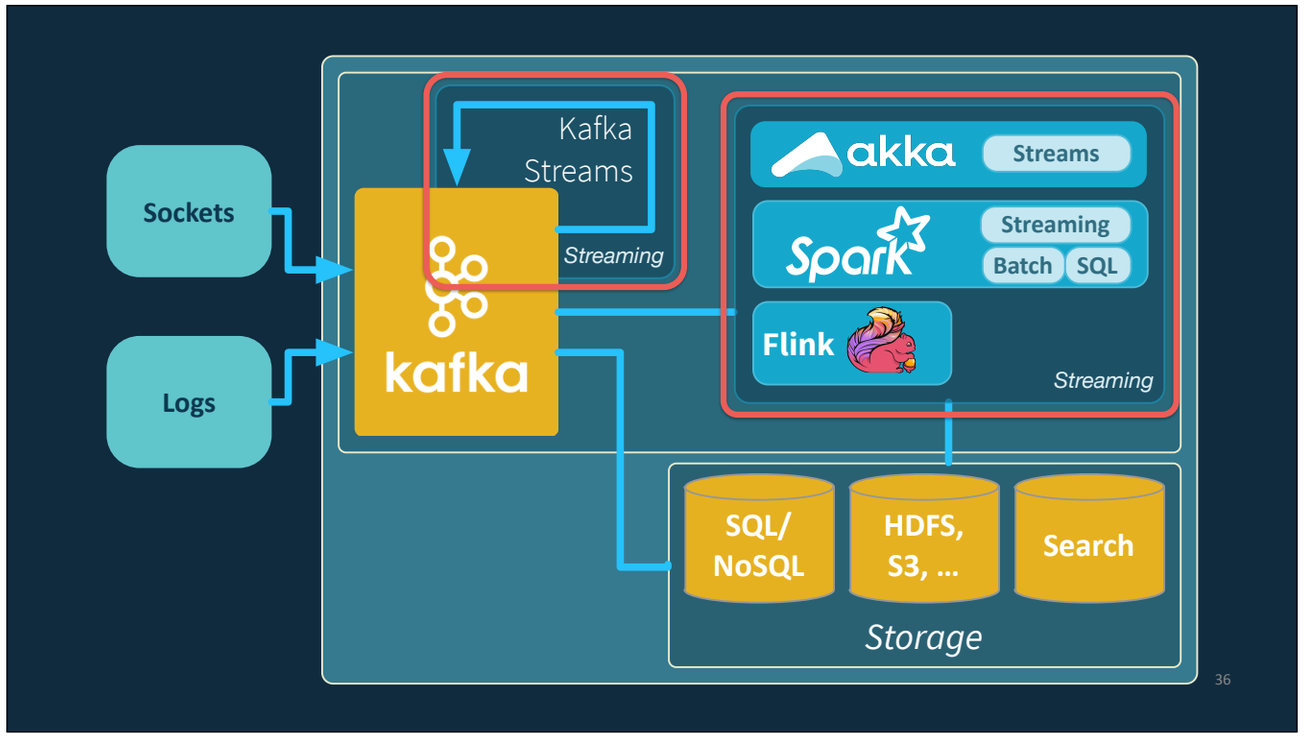
- Which kinds of data processing & analytics are required?
- How will this processing be done?
  - Individual processing of events?
  - Bulk processing of records?

Are you doing complex event processing (CEP)? Aggregations? ETL? Others?
CEP is (usually) best done with a tool that processes each event individually, whereas other kinds of data can be processed “en masse” and it’s more efficient to do so, (like joins and group-bys).

Streaming Tradeoffs (3/3)

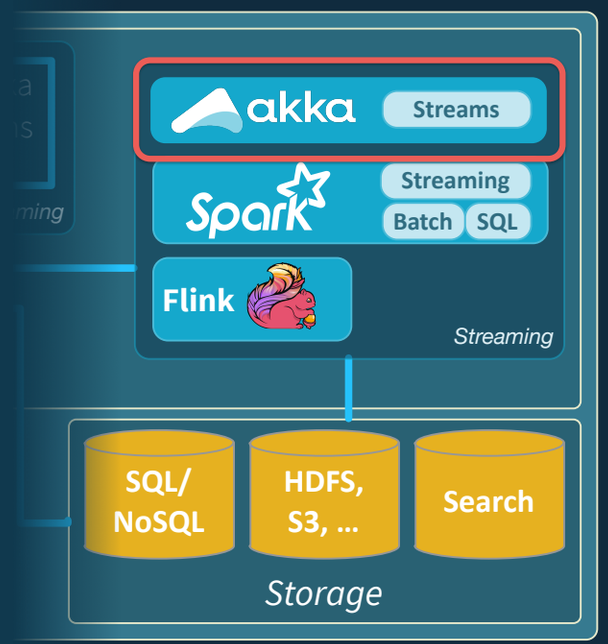
- Which tools and data sources/sinks must interoperate with your streaming tool?

Nothing lives in isolation. Data sources and sinks need to be connected with streaming engines and storage. Similarly, your microservices will interoperate with your data analytics.

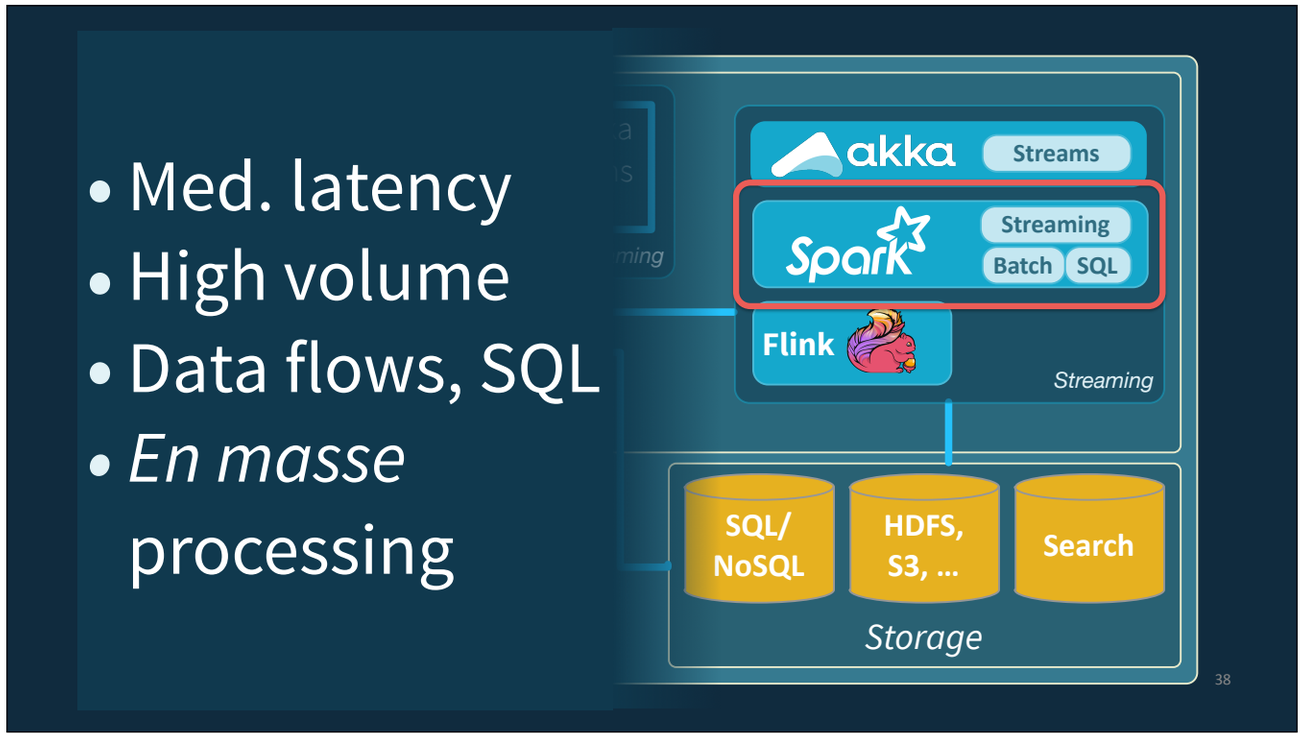


Let’s go back to the subset diagram and look at each of these tools.
Here are four of dozens(?) of possibilities for streaming engines. This four are chosen for their features that together meet all the needs we just discussed and they are reasonably mature projects with active communities: Akka Streams, Spark, Flink, and Kafka Streams.

- Low latency
- Low volume
- Complex flows
- Complex Event Processing

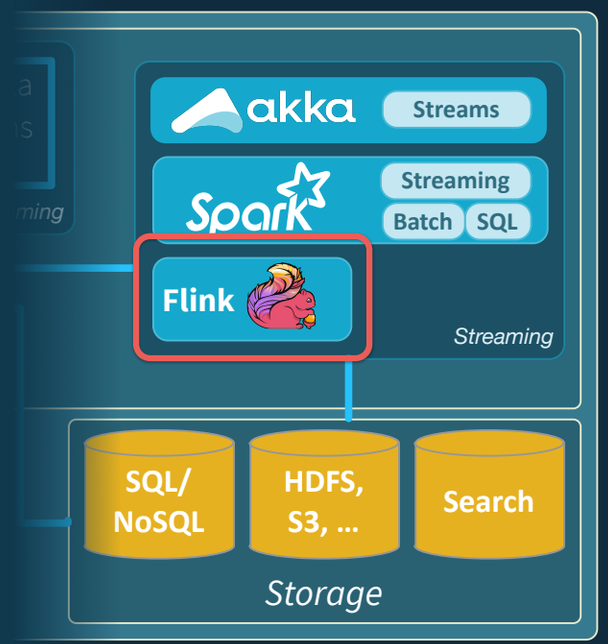


Akka is very low latency, optimized for excellent performance per event instead of high volume processing. You can do arbitrarily complex processing, including a sophisticated “flow graph” model. It is ideal for low-latency CEP (complex event processing). Akka Streams gives you intuitive dataflow semantics, so you don’t have to write lots of boilerplate when a streaming model fits your problem. Akka also provides rich integration with many other data sources/sinks, with RP-based micro services, etc.

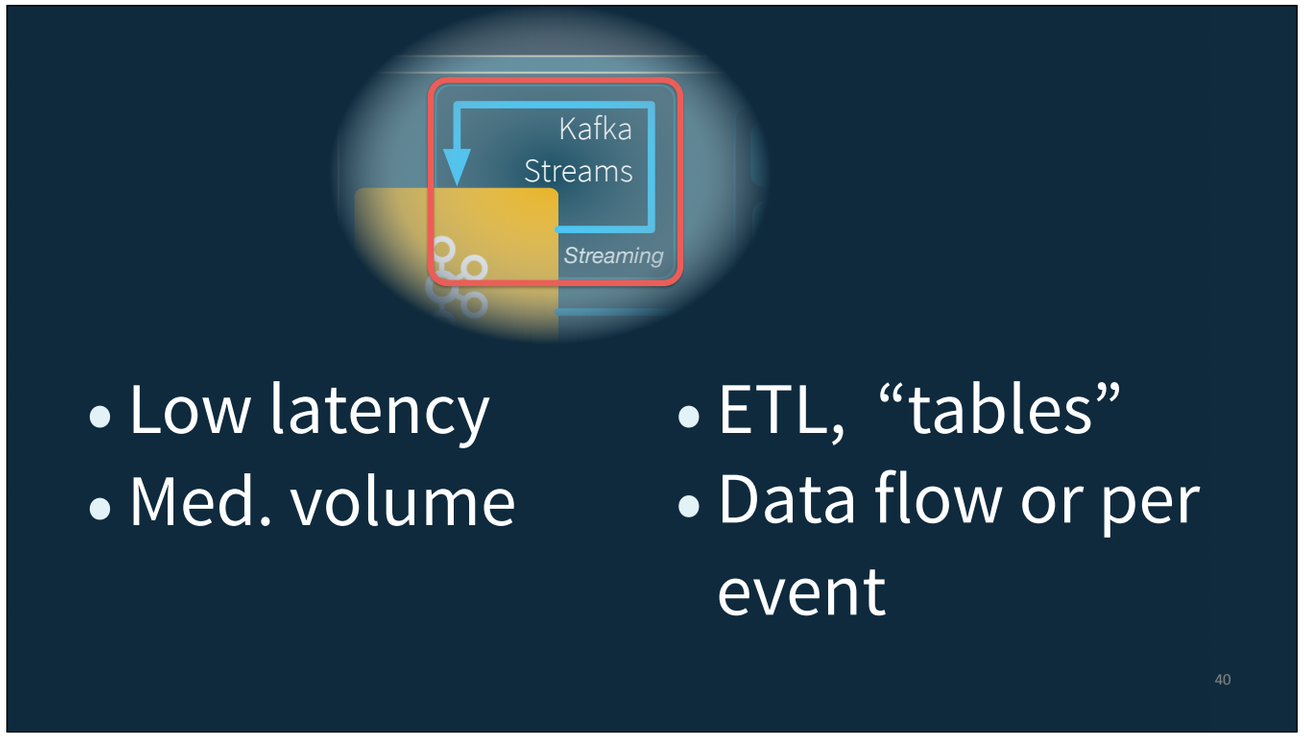


Spark has medium latency (~0.5 seconds and up), optimized for excellent, scalable performance at high volumes. The model is either data flows (think sequential processing nodes) or SQL queries. It is not designed for per-event processing, but “en masse” processing of records.

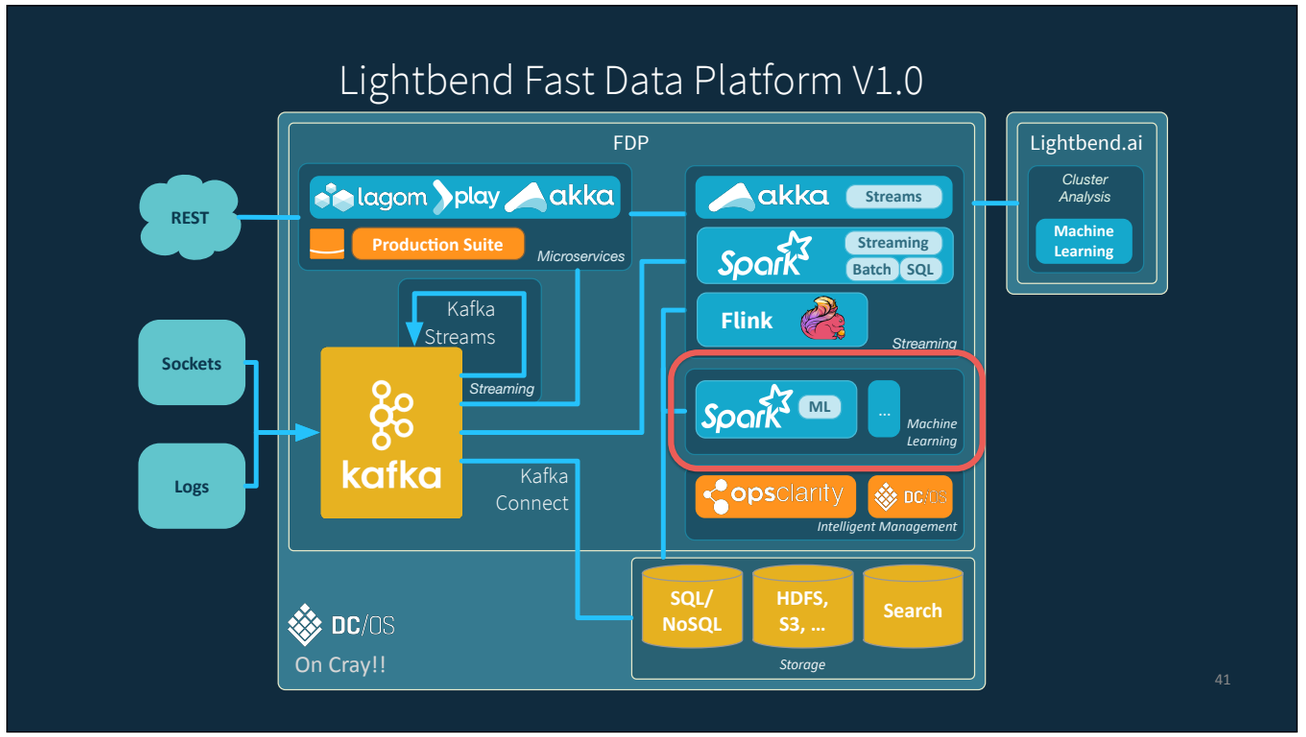
- Low latency
- High volume
- Data flows, correctness
- *En masse* processing



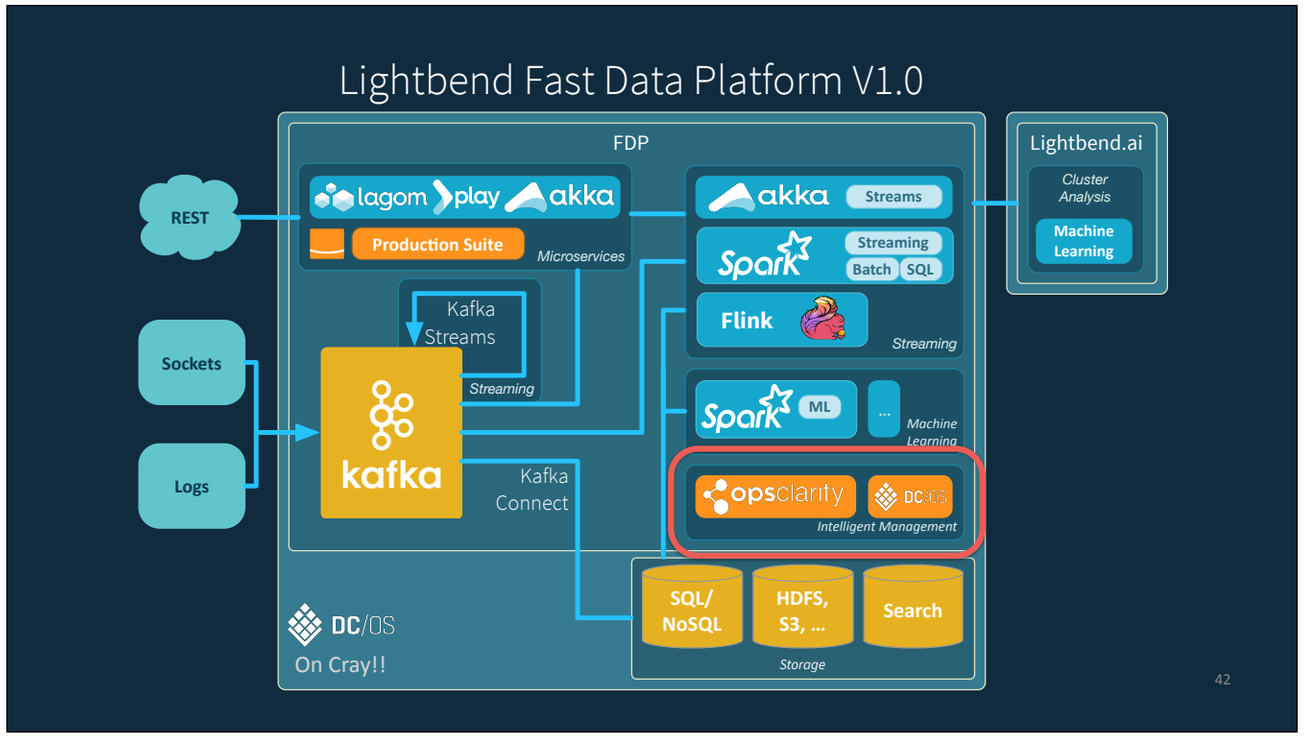
Flink contrasts mostly with Spark. It fixes two (current) limitations: 1) low latency instead of medium latency from the current mini-batch streaming model in Spark, 2) Flink provides state of the art semantics for more sophisticated stream processing, especially when high accuracy (as opposed to approximate values) is important. These semantics are defined by Apache Beam (a.k.a., Google Dataflow). Flink can run Beam data flows (The open source Apache Beam requires a third-party runner). Both Spark and Flink provide excellent, scalable performance at high volumes.



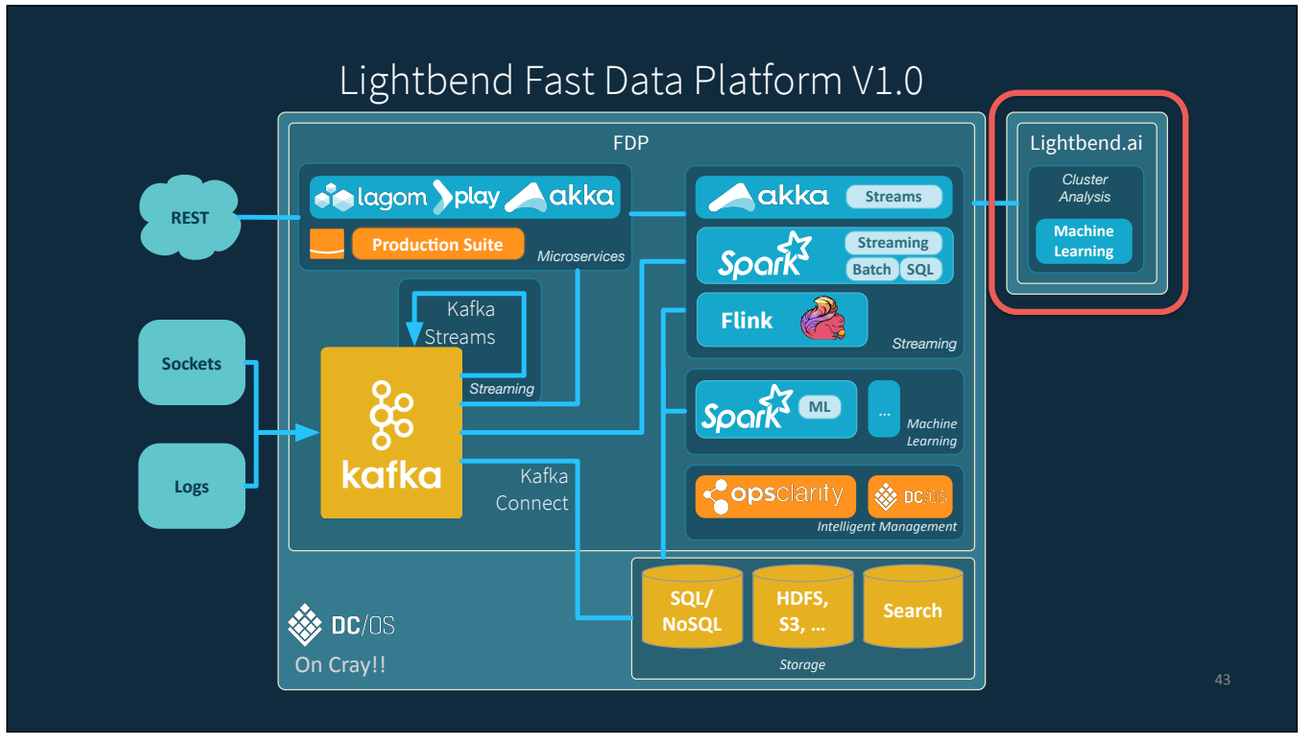
Kafka Streaming is a great “80%” tool. It doesn’t try to solve 100% of all problems, but what it does, it does very well. KS is focused on reading data in Kafka topics, processing it, and writing the results to new topics. It’s ideal for many common scenarios, such as ETL, but also supports running aggregations including the last seen values for keys (like DB tables work). Using the API, you write data flow code, but the implementation is more like a per-event processor.



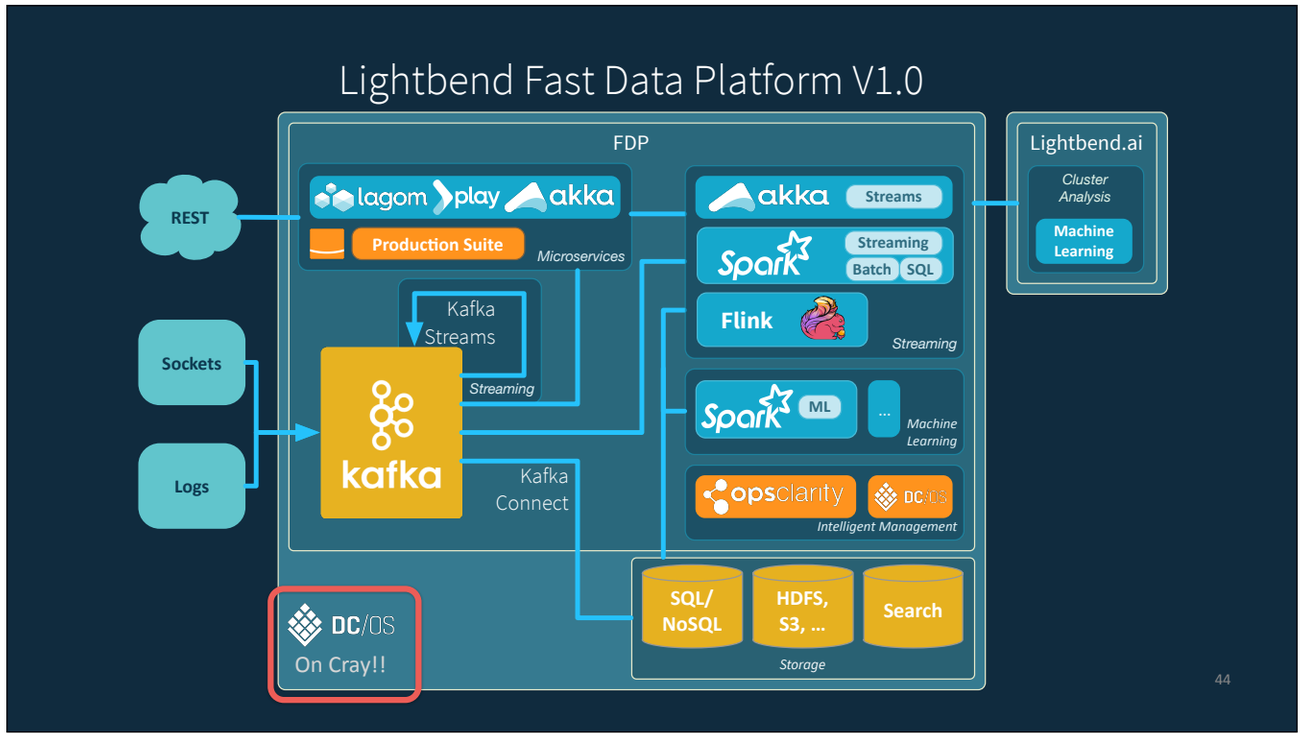
Moving on to the rest of the elements of FDP, Machine learning components in V1.0 include Spark's ML libraries and Intel's BigDL. Longer term, we will add other ML libraries to offer a full suite of options, as for our streaming libraries. We'll also focus on providing convenient solutions for common ML engineering problems, such as sharing models between streaming engines (e.g., training in minibatches in Spark and serving the models in Flink or Akka Streams).



A key benefit of FDP is integrated, cluster-wide monitoring powered by OpsClarity, which a cloud-based monitoring tool tailored for streaming pipelines with an intuitive, powerful UI. It complements the management and visualization capabilities of the DC/OS console. In addition application consoles that come with Spark, Flink, etc. are also part of the management suite, but the most innovative component is …



... cluster analysis through our machine-learning based **Lightbend.ai** service, which analyzes cluster telemetry to train models for anomaly detection, predictive maintenance, autoscaling, and other capabilities designed to keep your FDP clusters running reliably with minimal hands-on maintenance.

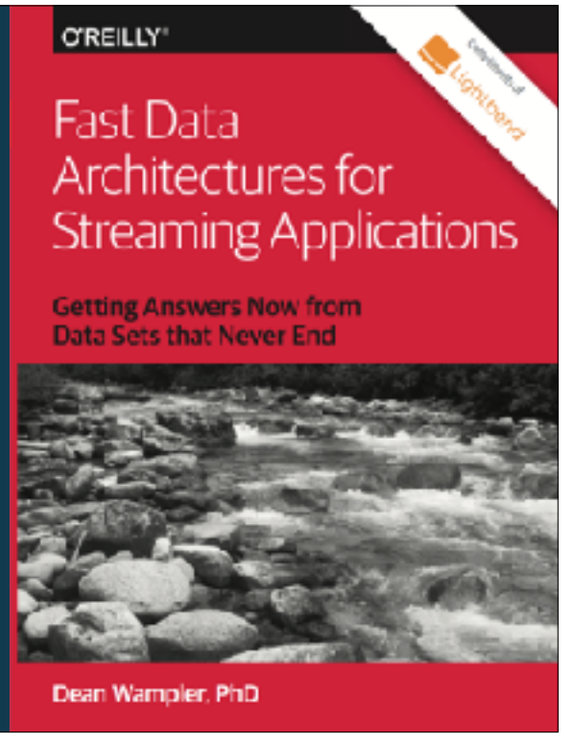


FDP leverages DC/OS, the Mesos-based, state of the art cluster resource and application manager, deployable on Cray!

For More Information

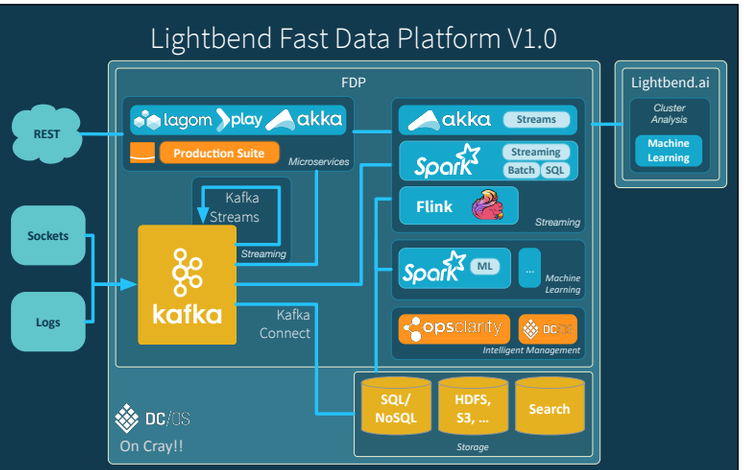
Free, as in 🍺

bit.ly/lightbend-fast-data



My report on Kafka and the streaming engines from a more general perspective and in more depth. Not specific to FDP.

For More Information



lightbend.com/fast-data-platform

For more information on the Fast Data Platform.