FN1987 – Using Splunk Data Stream Processor as a Streaming Engine for Apache Kafka

Thor Taylor and Adam Lamar

Thor Taylor
Director of Product – Stream Processing | Splunk

Adam Lamar
Principal Software Engineer | Splunk

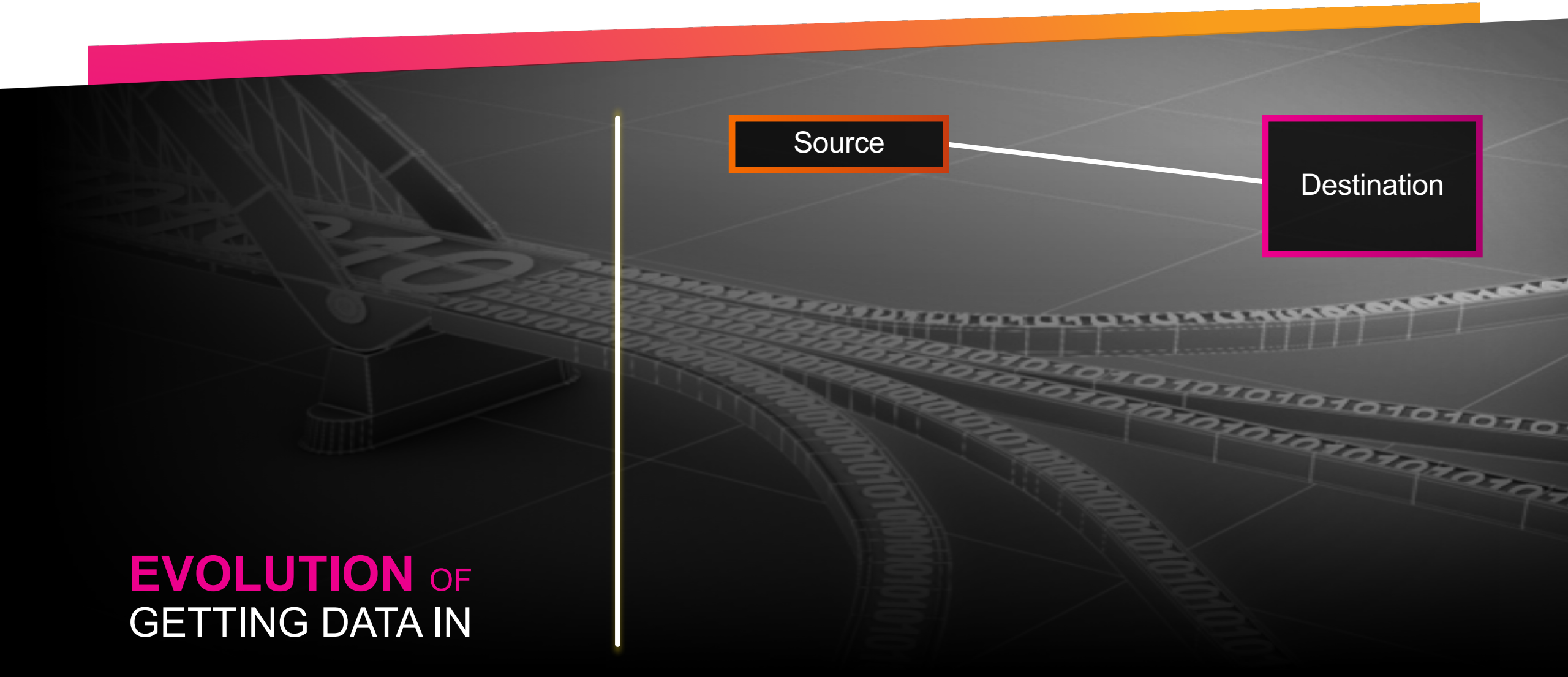
splunk> .conf19

Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or plans of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results may differ materially. The forward-looking statements made in the this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, it may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements made herein.

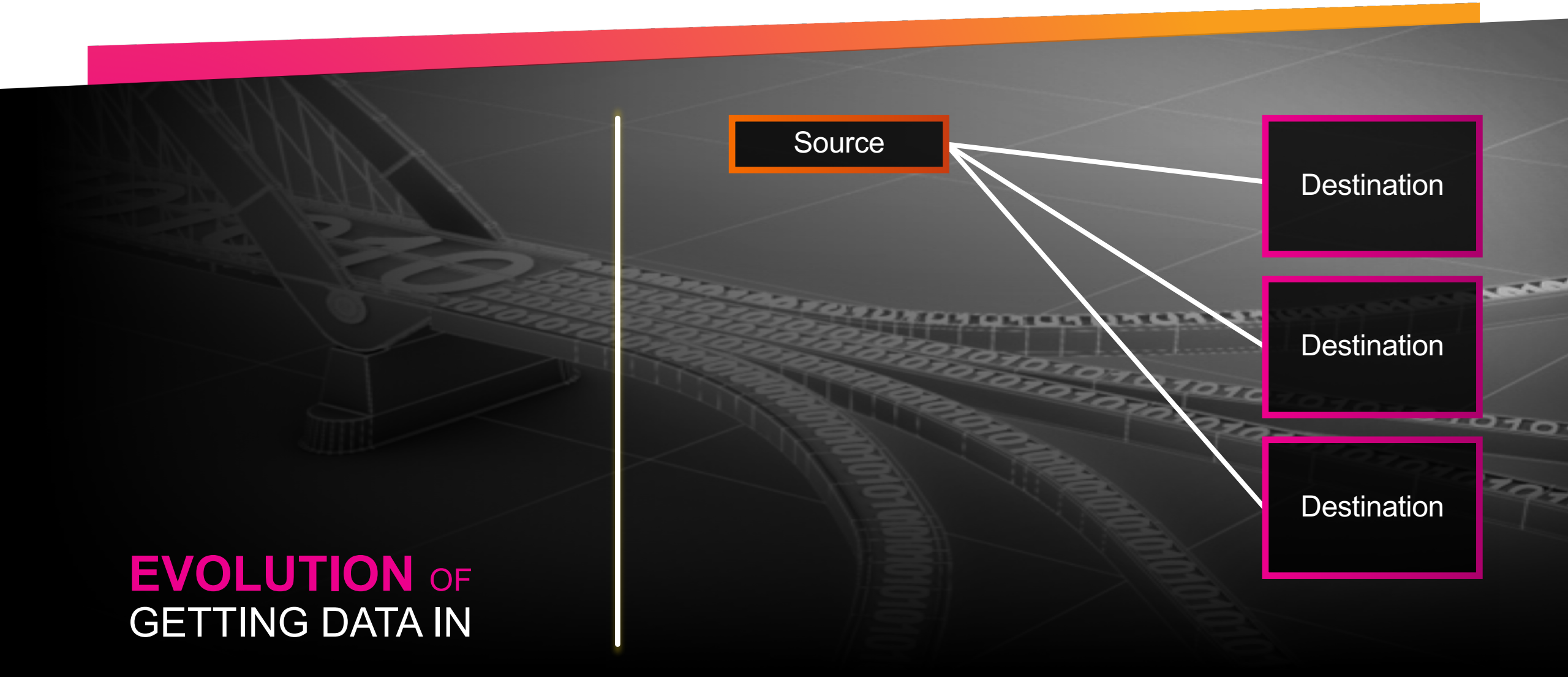
In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only, and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionalities described or to include any such feature or functionality in a future release.

In the beginning it was simple



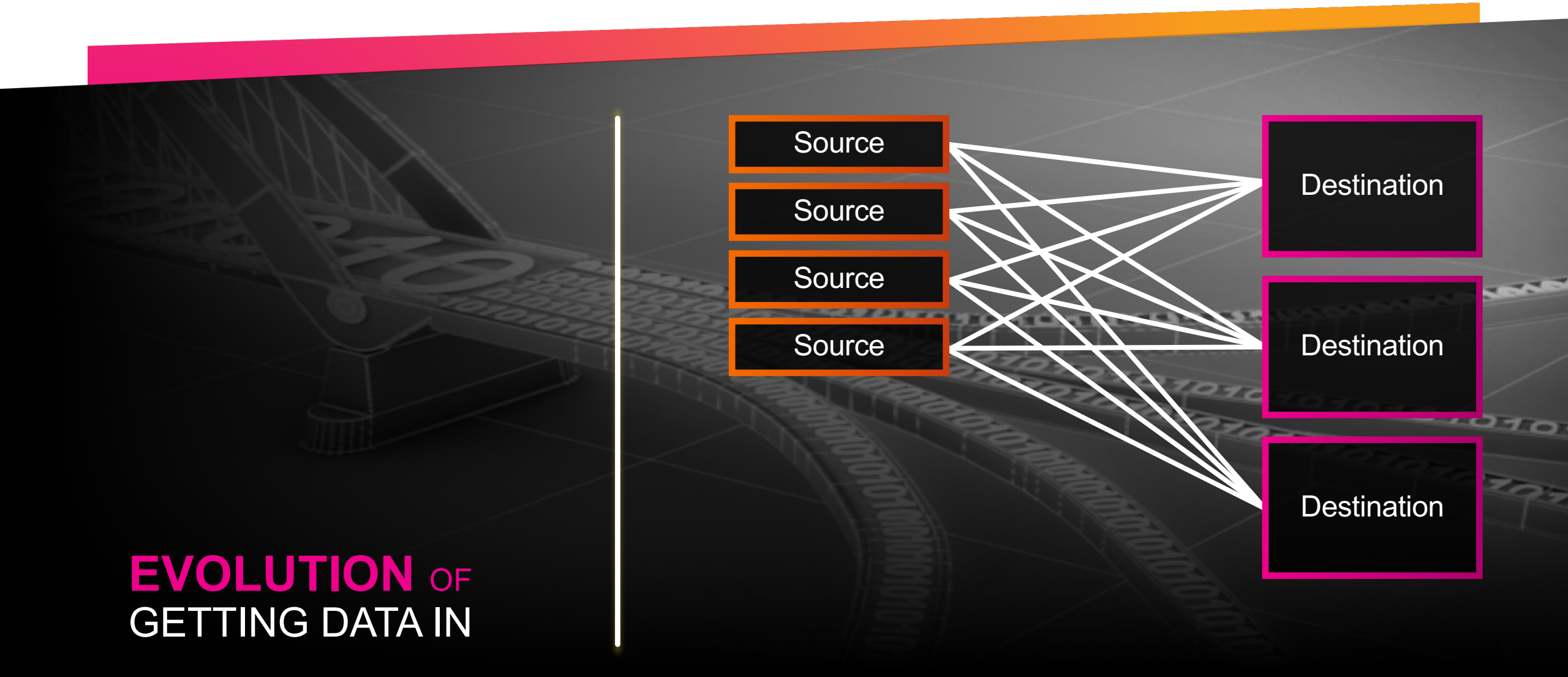
EVOLUTION OF
GETTING DATA IN

Then more systems needed data



EVOLUTION OF
GETTING DATA IN

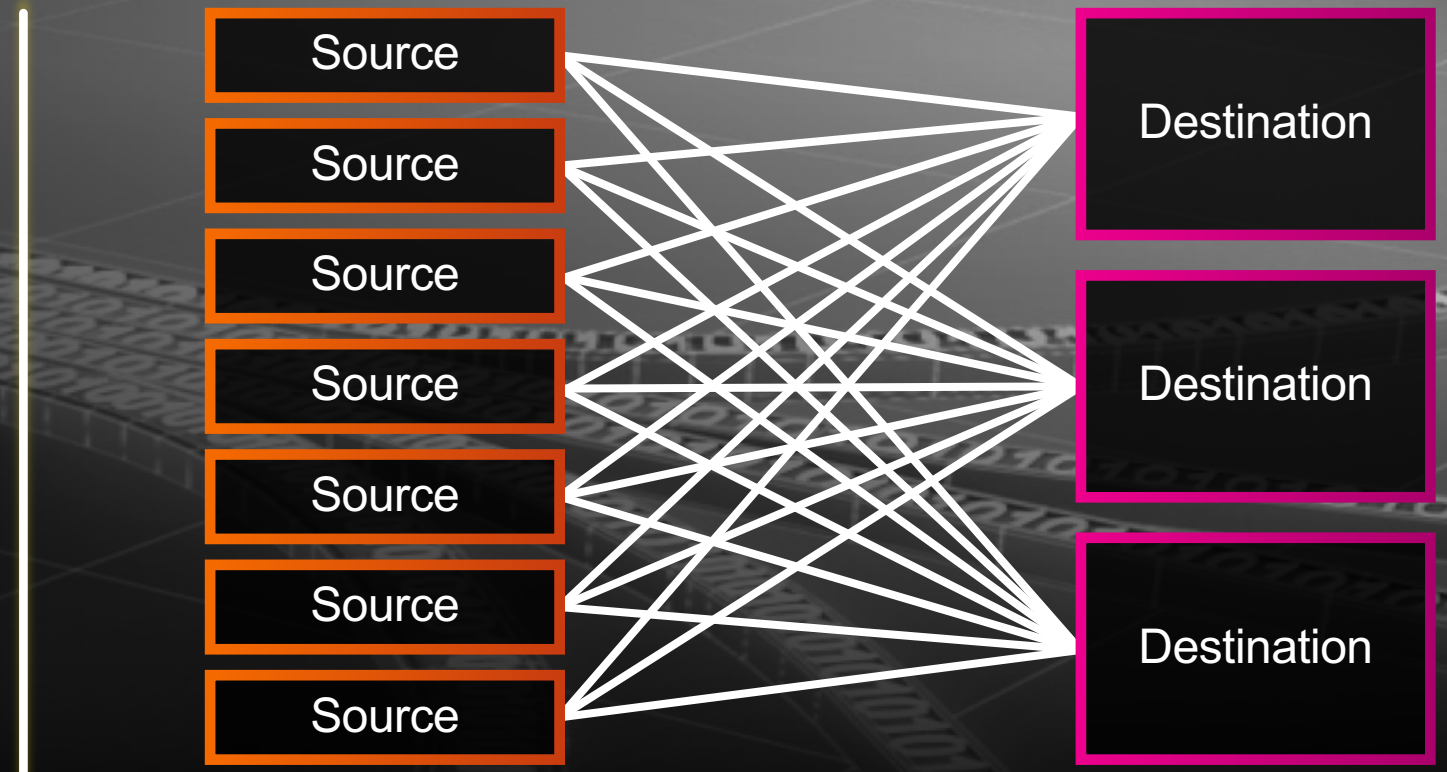
Then more data was produced



EVOLUTION OF
GETTING DATA IN

Creating complexity

- How can we **view** and **monitor** the health of the data transfer lifecycle?
- We need to leverage the **same data** in **multiple systems**, how do we do this without sending it to all systems?
- How do we **scale** and **guarantee delivery**?



EVOLUTION OF
GETTING DATA IN

Without a Message Bus

Current behavior

- M data producers
- N data consumers
- Produces M*N data paths

Every producer must be aware of every consumer

- Configuration challenge
- New destinations require configuration management solution

Leads to complex and brittle ingestion architectures

Message bus solves this problem

Why Kafka Specifically?

Established leader in event-processing architectures

Kafka grows well with your needs

- Partitioning
- Authentication
- Role-based access
- Configurable retention
- Scales independent of producers and consumers
- Active community

What kafka doesn’t do

- Process messages faster
- Provide free lunch

Why Kafka Specifically?

Topic-based

- Group similar data together

Durability

- Producers can assume data persistence after send
- Reduction of producer complexity

Replication

- Data received can be replicated before ack

Producer/consumer decoupling

- Scale producers and consumers independently
- Producers aren’t concerned with final data destination
- Producers can respond elastically to increase in capacity

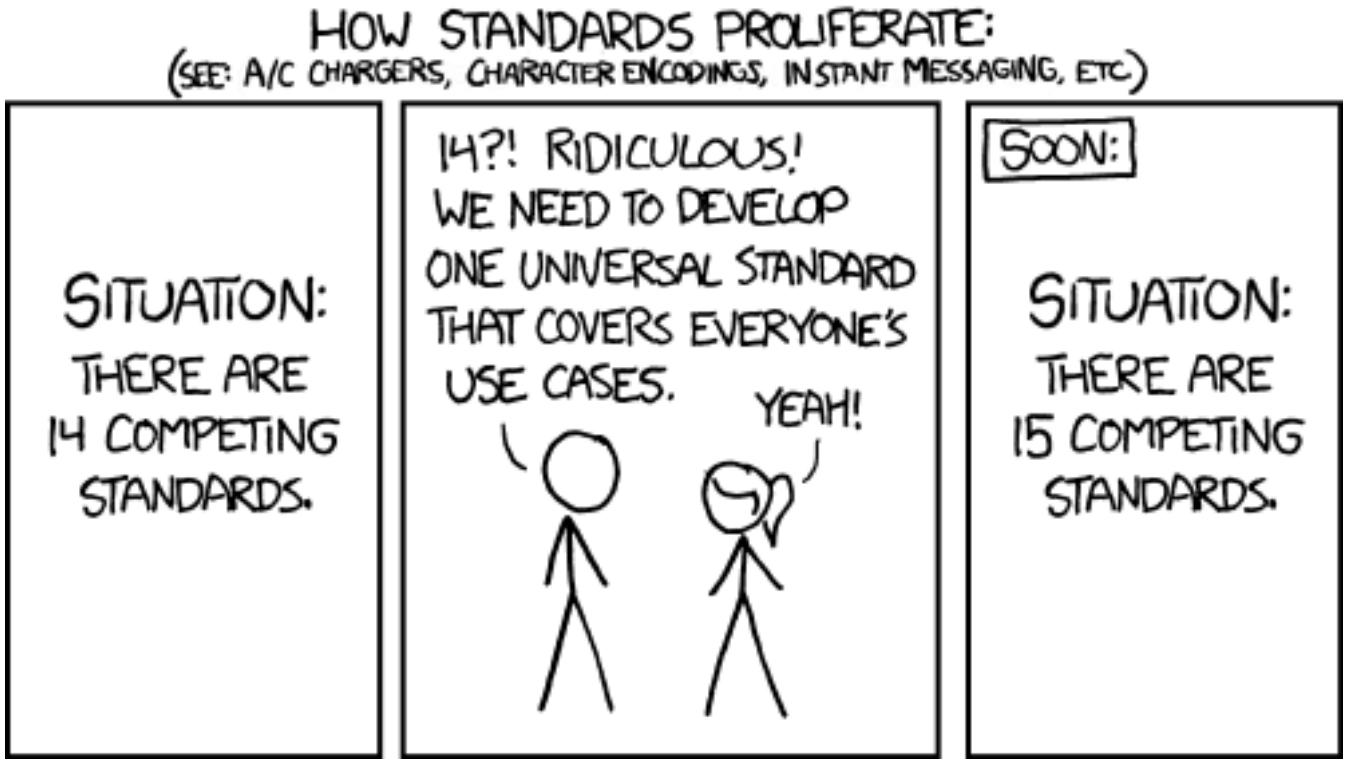
Which required a solution

- Gain **visibility** and **control** of data while its in motion.
- Send data to **any location** using a **publish/subscribe** architecture.
- **Scale** elastically, vertically or horizontally, based **on data needs**

EVOLUTION OF
GETTING DATA IN

Not all data is created equally

But all sources **WILL** need to send to many destinations

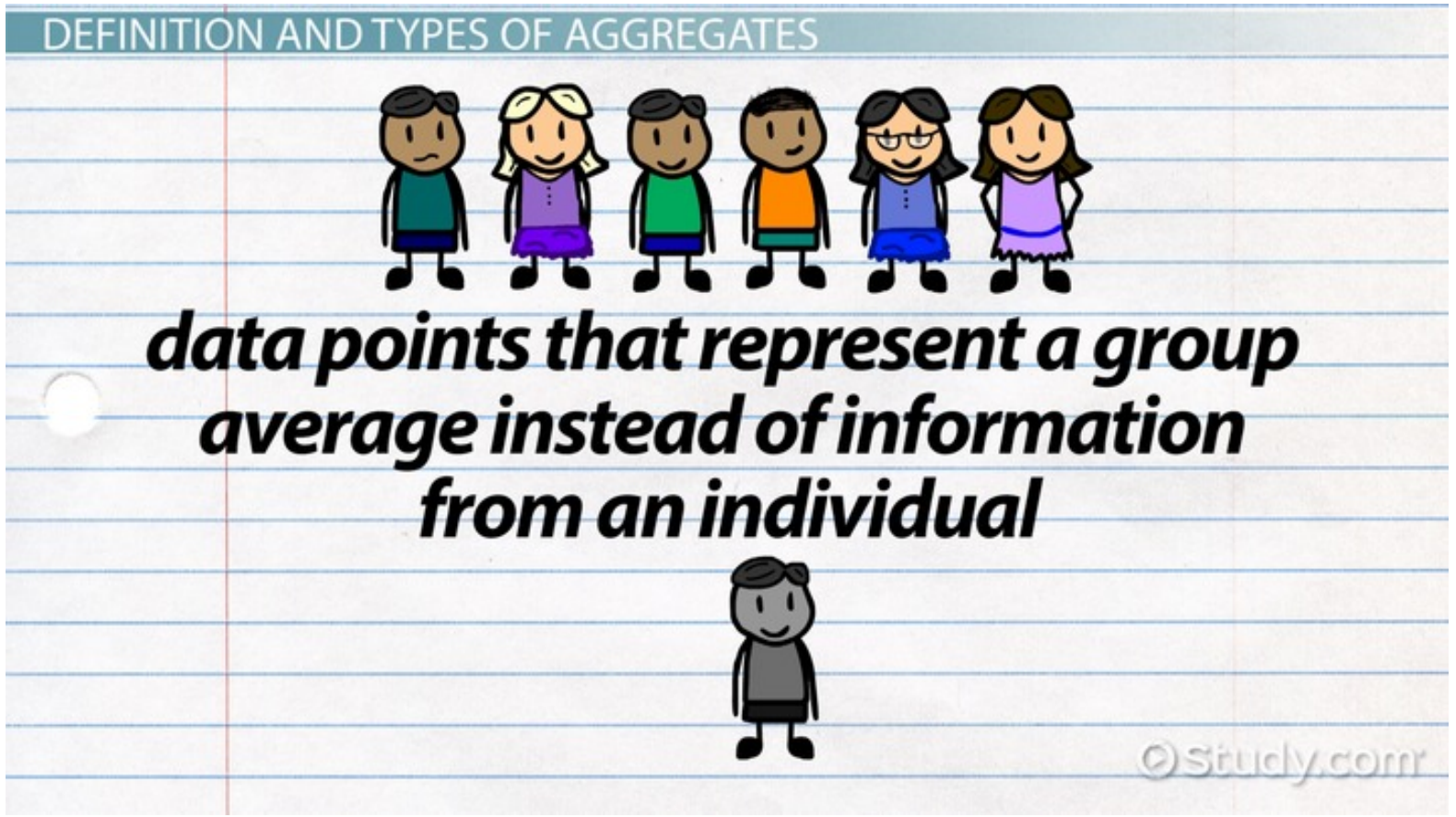


Some Info is Perishable

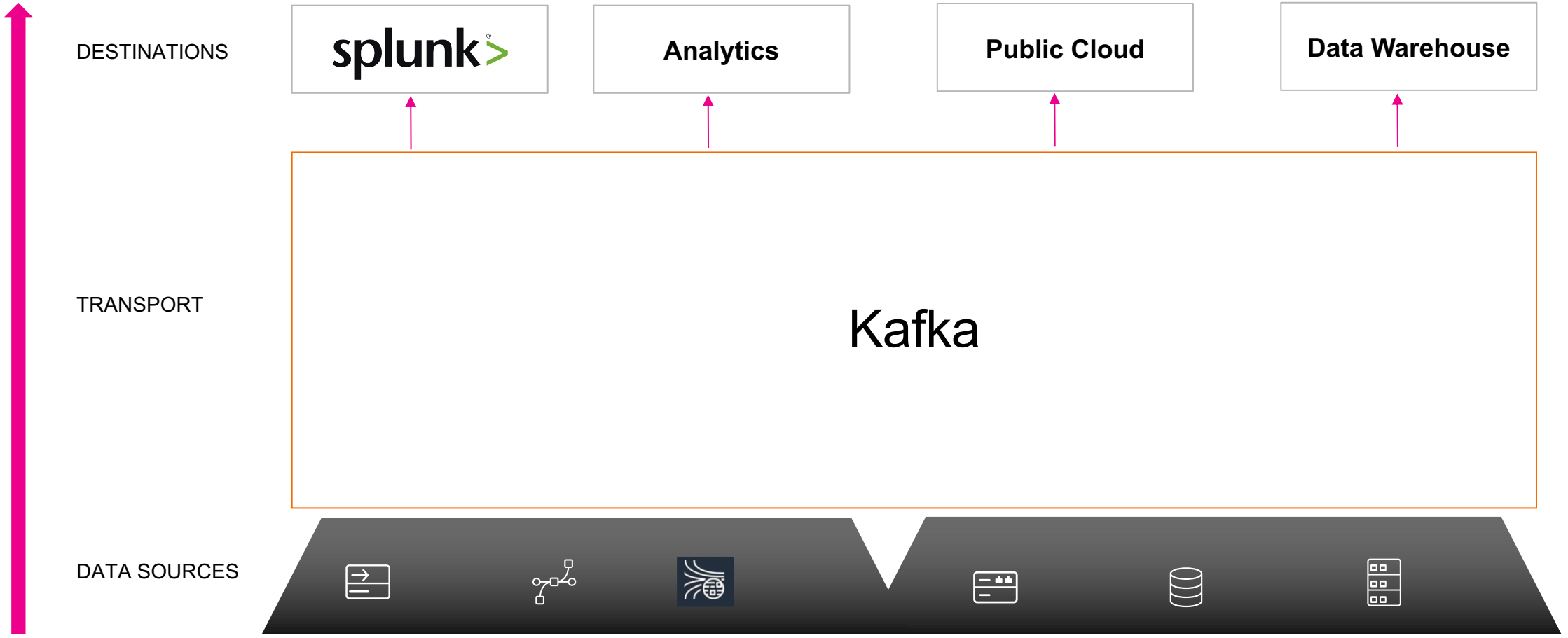
Sometimes it helps to be proactive rather than reactive

Sometimes we just need a summary

See the forest not the trees



Streaming Architecture



More Data More Problems

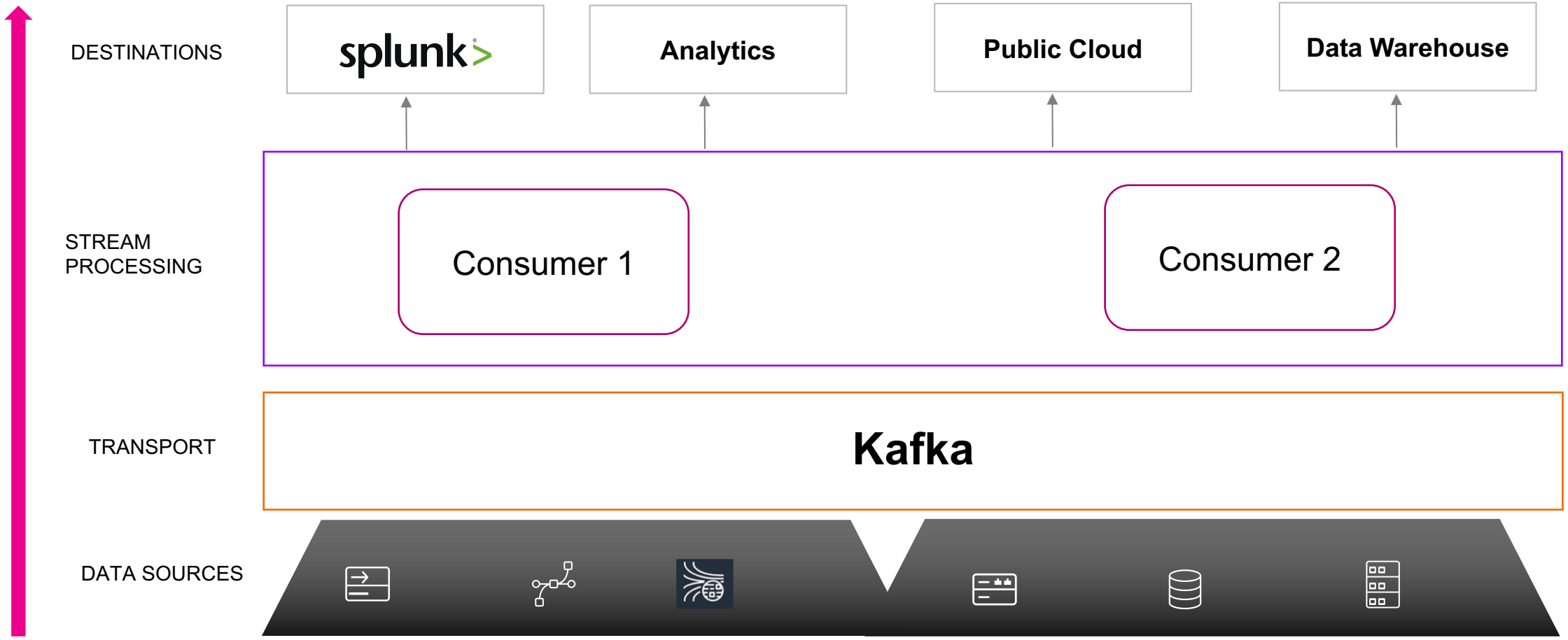
Kafka provides higher level abstractions

- Topics and data movement
- Improved data delivery
- Improved tooling

More consumer complexity

- Offset management
- State and configuration storage
- Schema management
- Consumers usually send the data elsewhere
- Not all consumers are created equal

Streaming Architecture



Consumer Logic

Business logic

- Routing
- Aggregation
- Batching
- Transformation
- Enrichment
- Redaction
- Compliance

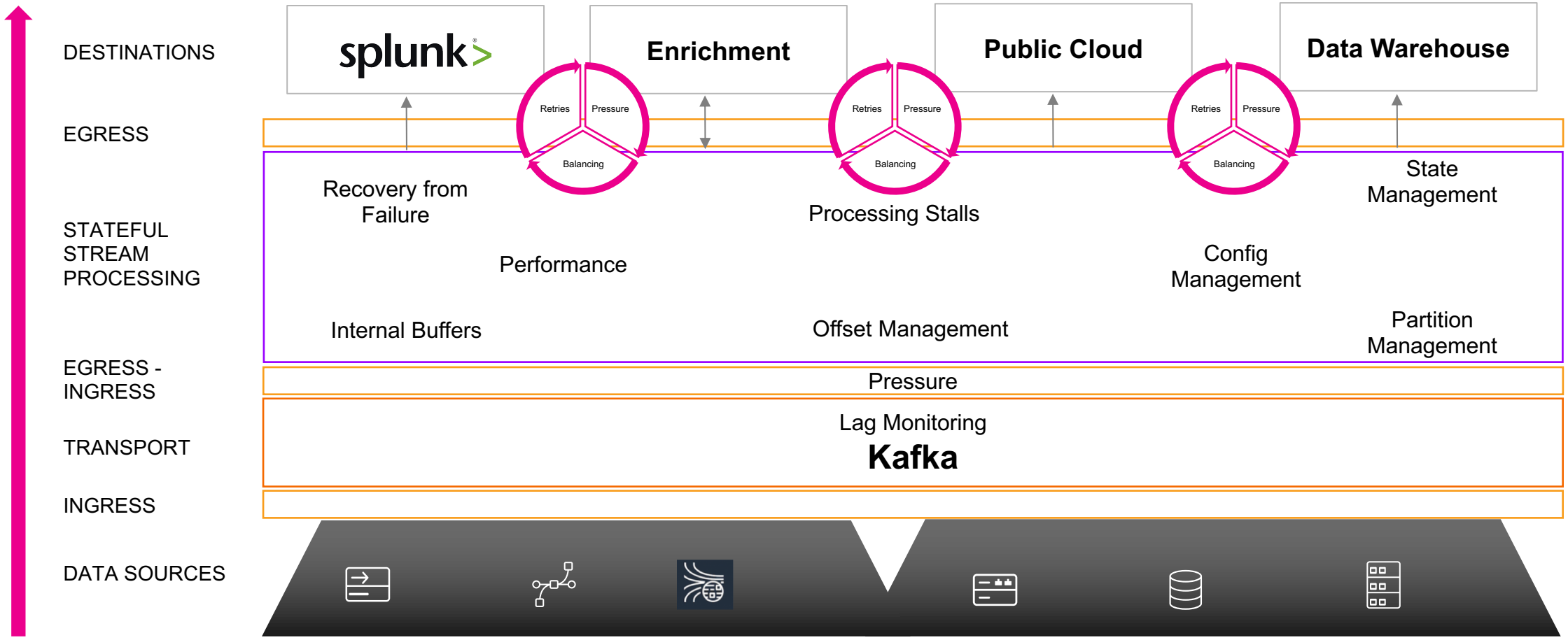
Business needs vary between orgs and change frequently

Technical challenge to process the stream in a robust manner

- Likelihood of data loss

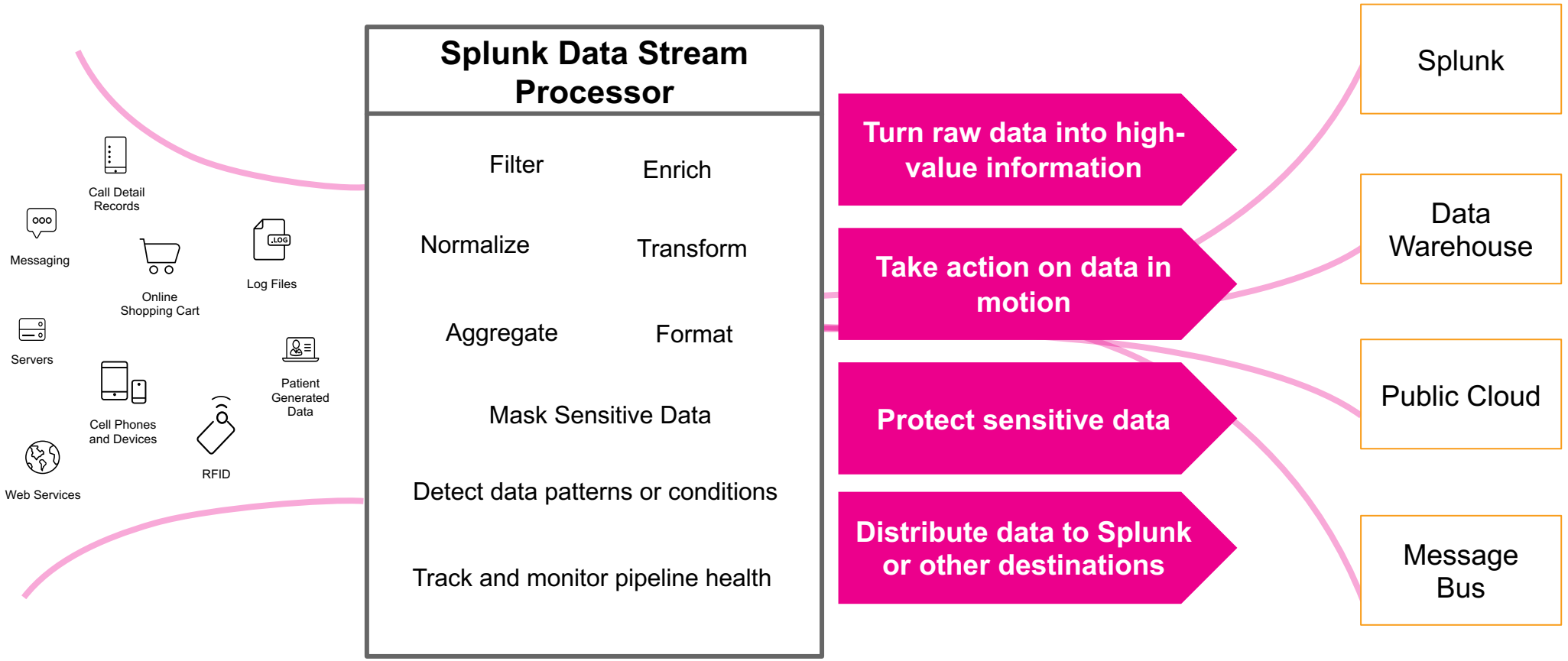
Stateful Stream Processing

Fault Tolerant Processing



Splunk Data Stream Processor

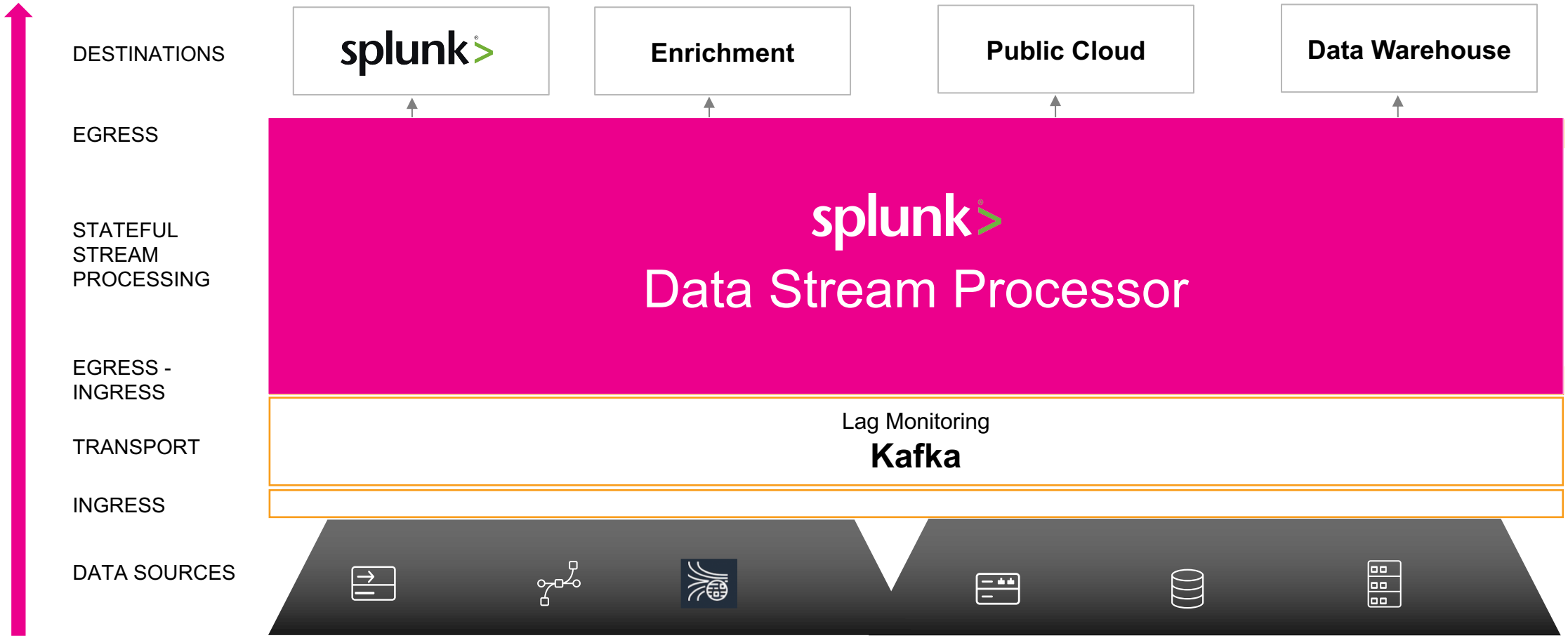
A real-time streaming solution that collects, processes, and delivers data to Splunk and other destinations in milliseconds



splunk> .conf19

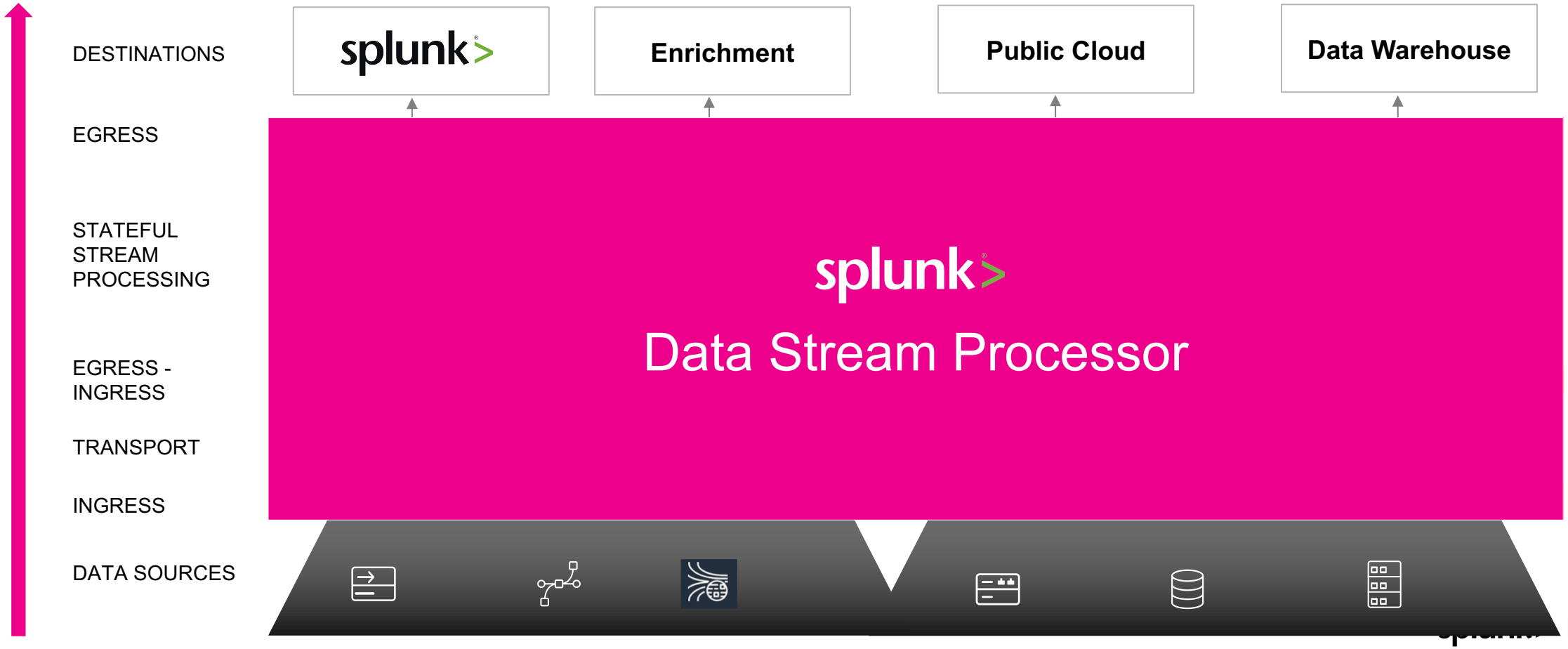
DSP for Kafka

Fault Tolerant Processing



DSP Standalone

Fault Tolerant Processing



Get Started Today!

Hardware Requirements

- Minimum Node Requirement
  - CPU: 8 core (16 recommended)
  - Memory: 64GB (128GB recommended)
  - Network: 10GBPS
  - Storage: 1TB
- Minimum 5 Node Cluster

Supported Data Sources

- Kafka, Kinesis, S3, CloudTrail, Event Hubs, REST APIs, Splunk (Universal Forwarder, Heavy Weight Forwarder, Http Event Collector)

Supported Destinations

- Kafka, Kinesis,Splunk

Demo Time

splunk> .conf19

.conf19

splunk>

Thank You!

Go to the .conf19 mobile app to

RATE THIS SESSION