



PUBLIC SECTOR
SUMMIT
BRUSSELS

ANT201

Everything You Need to Know About Big Data: From Architectural Principles to the Best Practices

Manos Samatas
Big Data and Analytics Solutions
Architect
AWS

What to expect from the session

Big data challenges

Architectural principles

How to simplify big data processing

What technologies should you use?

Why?

How?

Reference architecture

Design patterns

Customer examples

Types of big data analytics

Batch/
interactive



Stream
processing

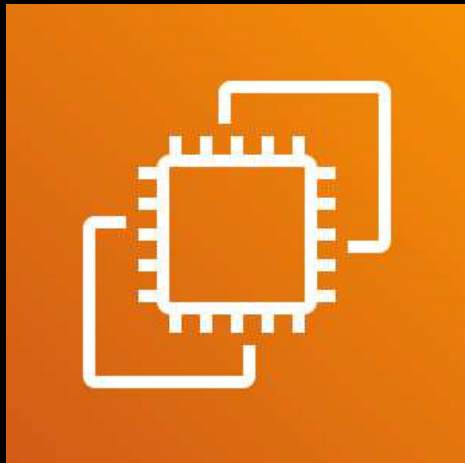


Machine
learning



Delivery of big data services

Virtualized



Managed
services



Serverless/
clusterless/
containerized



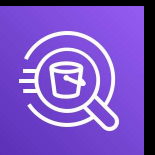
Plethora of tools



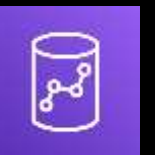
AWS Glue



Amazon
Elasticsearch
Service



Amazon Athena



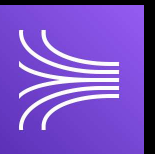
Amazon Redshift



Amazon Managed
Streaming for Kafka



Amazon
CloudSearch



Amazon Kinesis



Amazon EMR



Amazon Aurora



Amazon QuickSight



Data Pipeline



AWS Lake Formation



Amazon DocumentDB



Amazon Neptune

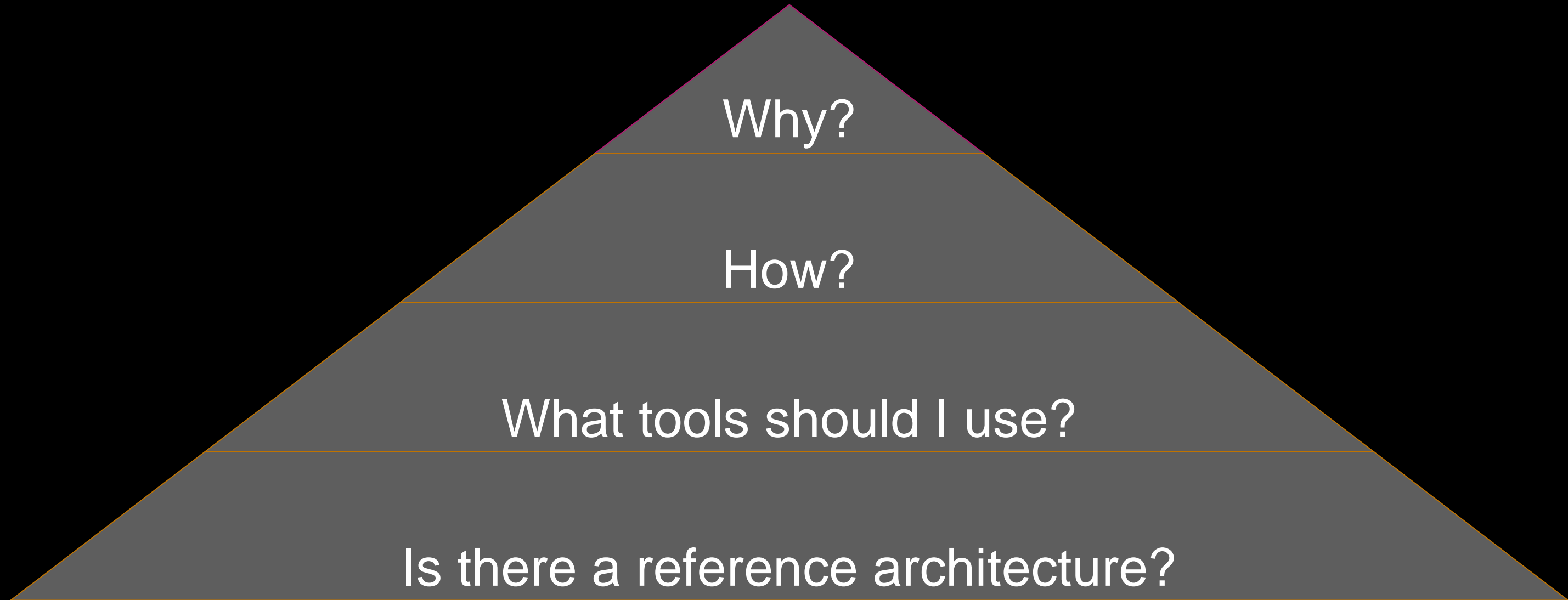


ElastiCache



DynamoDB

Big data challenges



Architectural principles

Build decoupled systems

- Data → Store → Process → Store → Analyze → Answers

Use the right tool for the job

- Data structure, latency, throughput, access patterns

Leverage managed and serverless services

- Scalable/elastic, available, reliable, secure, no/low admin

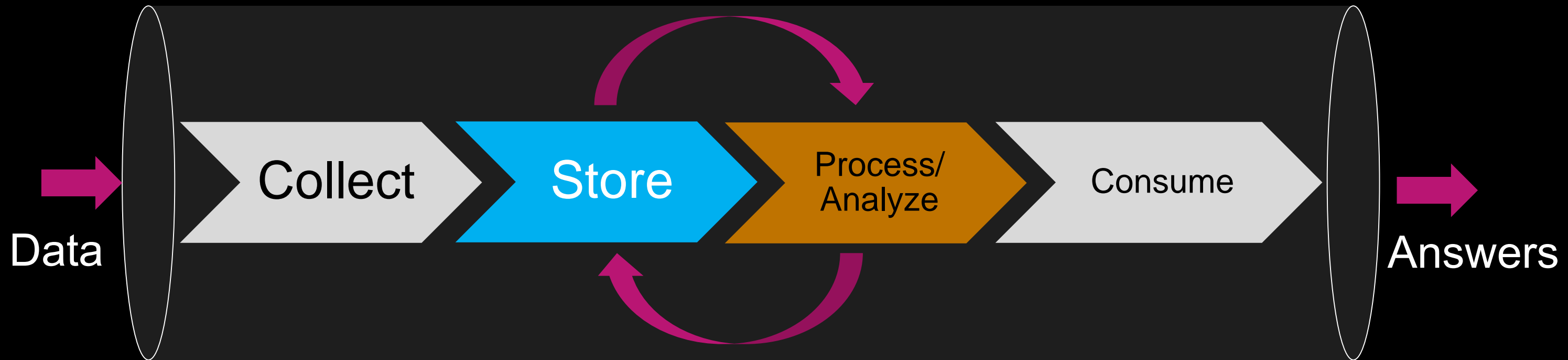
Use event-journal design patterns

- Immutable datasets (data lake), materialized views

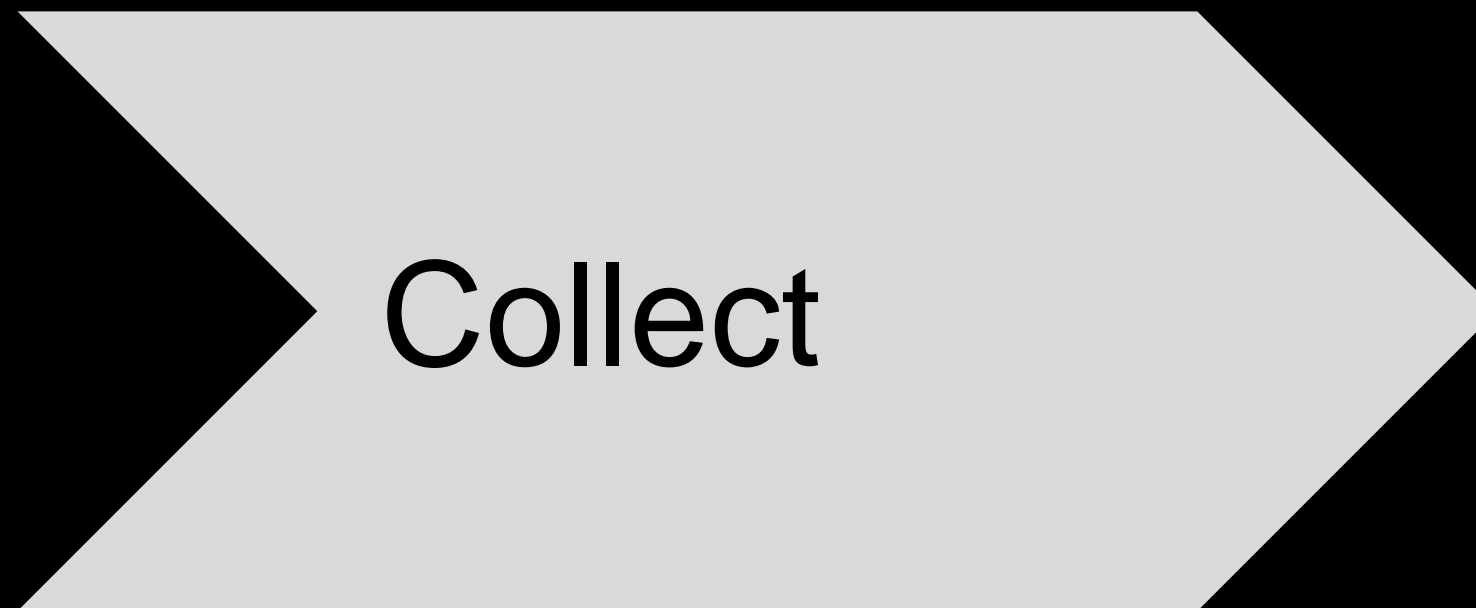
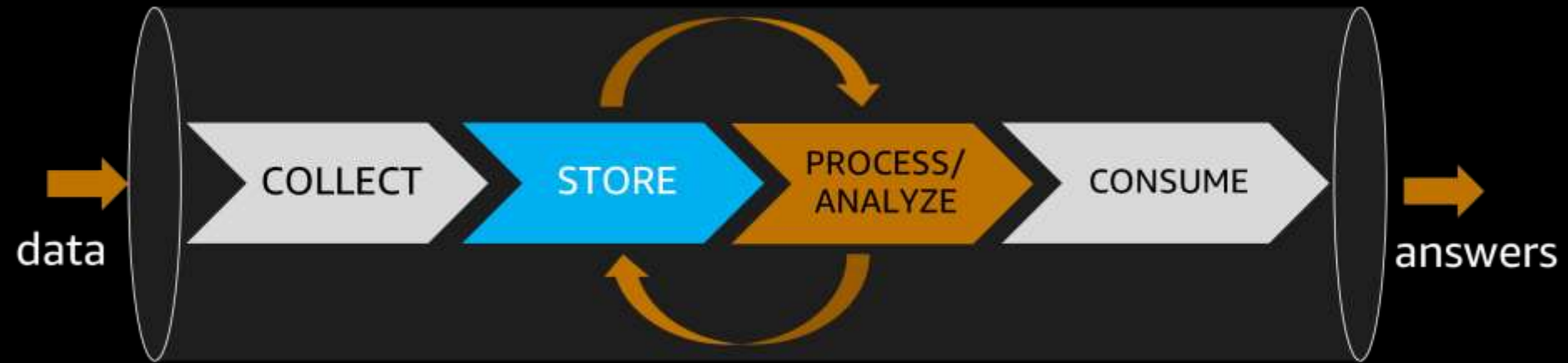
Be cost-conscious

- Big data ≠ big cost

Simplify big data processing



Time to answer (latency)
Throughput
Cost



Collect

Type of data

Data structures
Database records

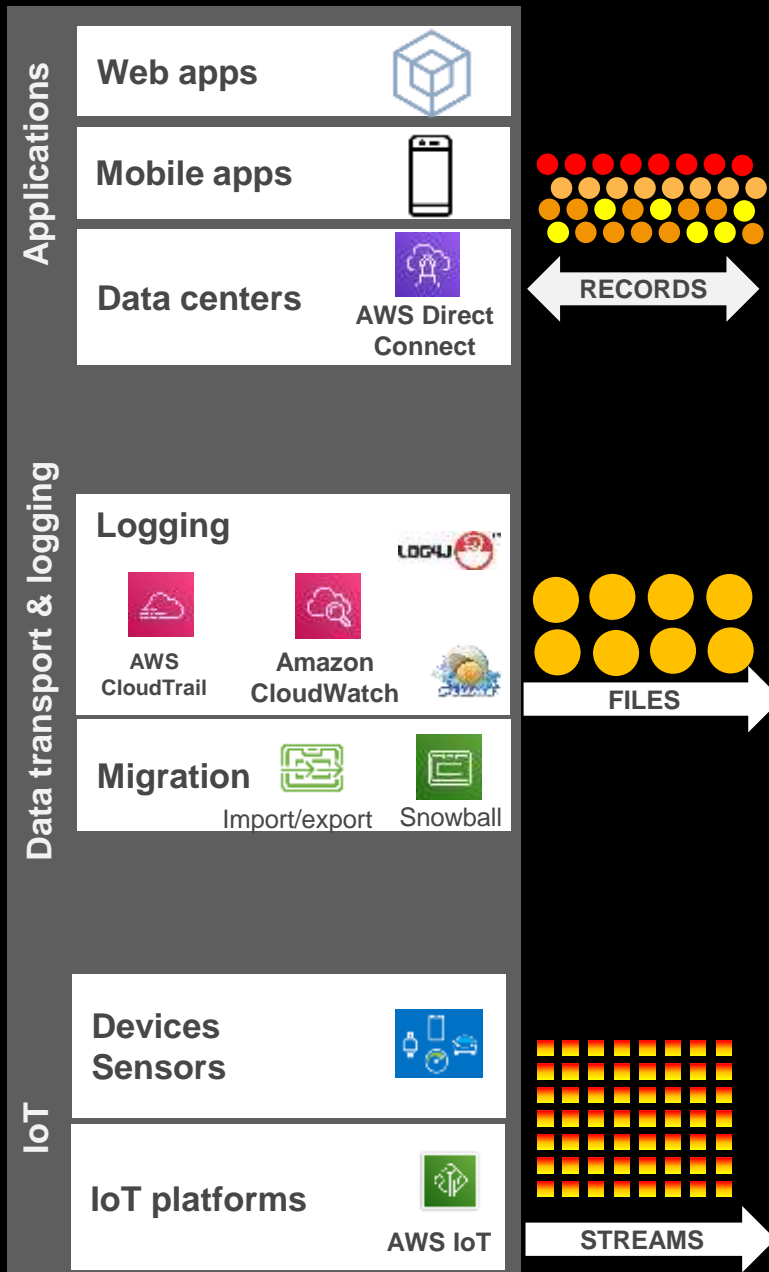
Transactions

Media files
Log files

Files/objects

Data streams

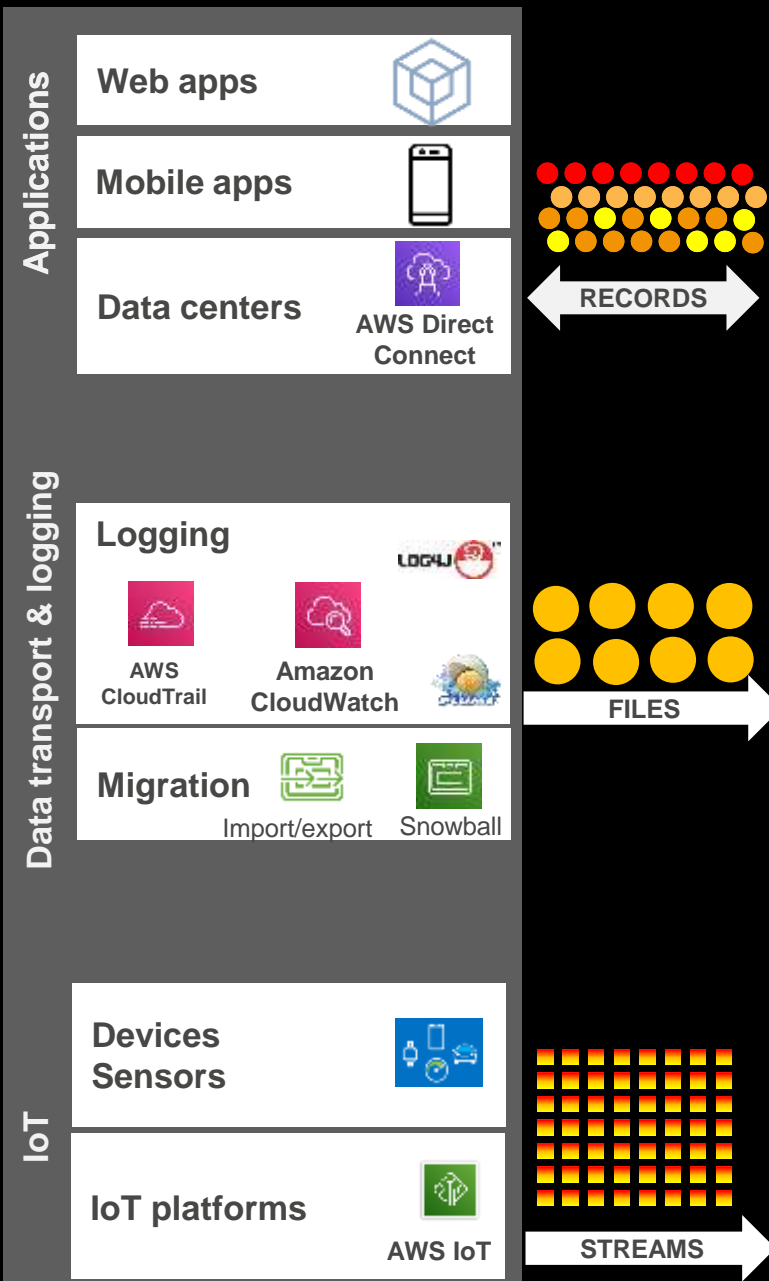
Events



Collect

Type of data

Store



Data structures
Database records

Media files
Log files

Data streams

In-memory

NoSQL

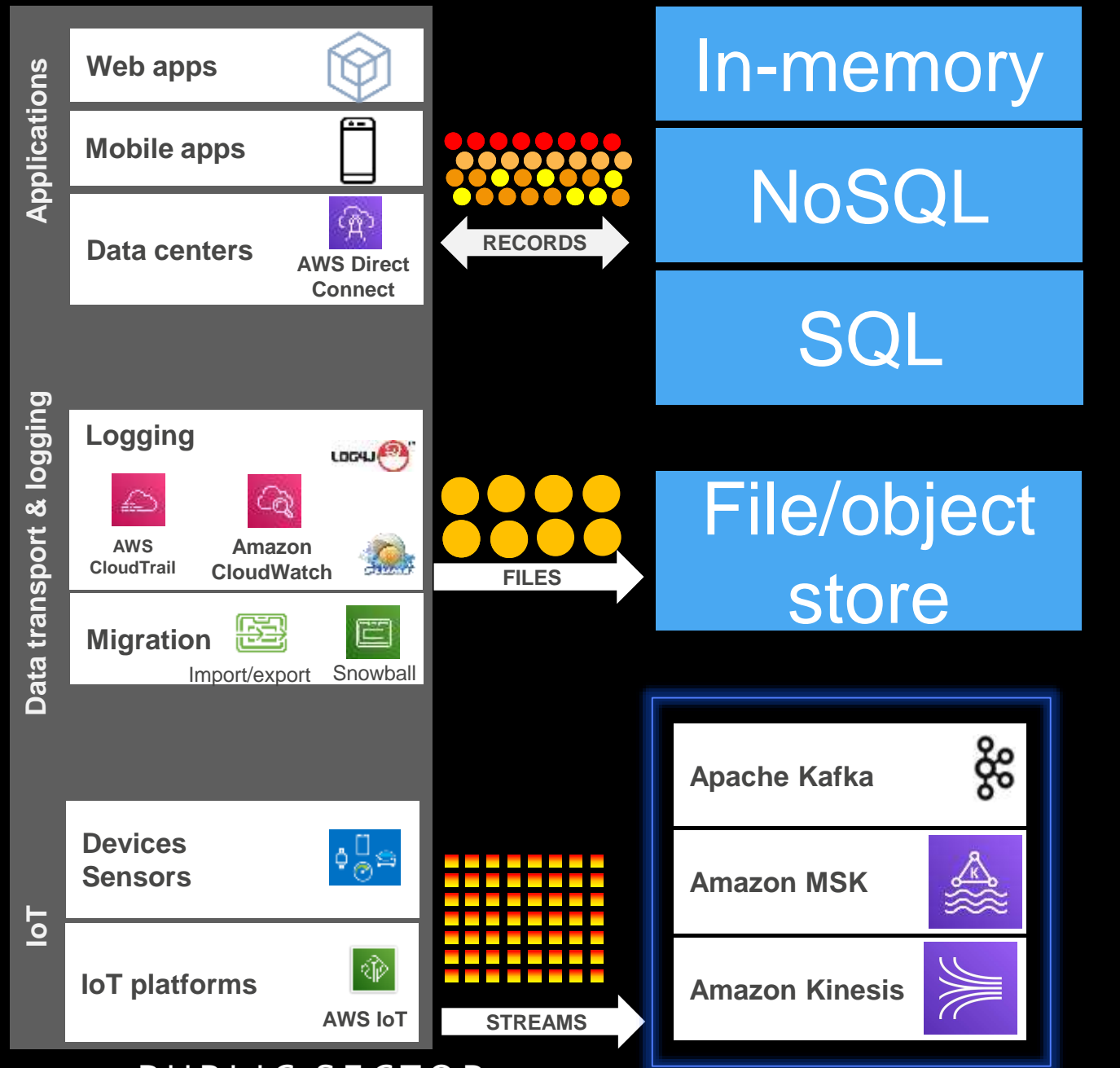
SQL

File/object
store

Stream
storage

Collect

Store



Stream storage

Apache Kafka

- High throughput distributed streaming platform

Amazon Kinesis

- Managed stream service

Amazon MSK - (in Preview)

- Managed Service for Kafka

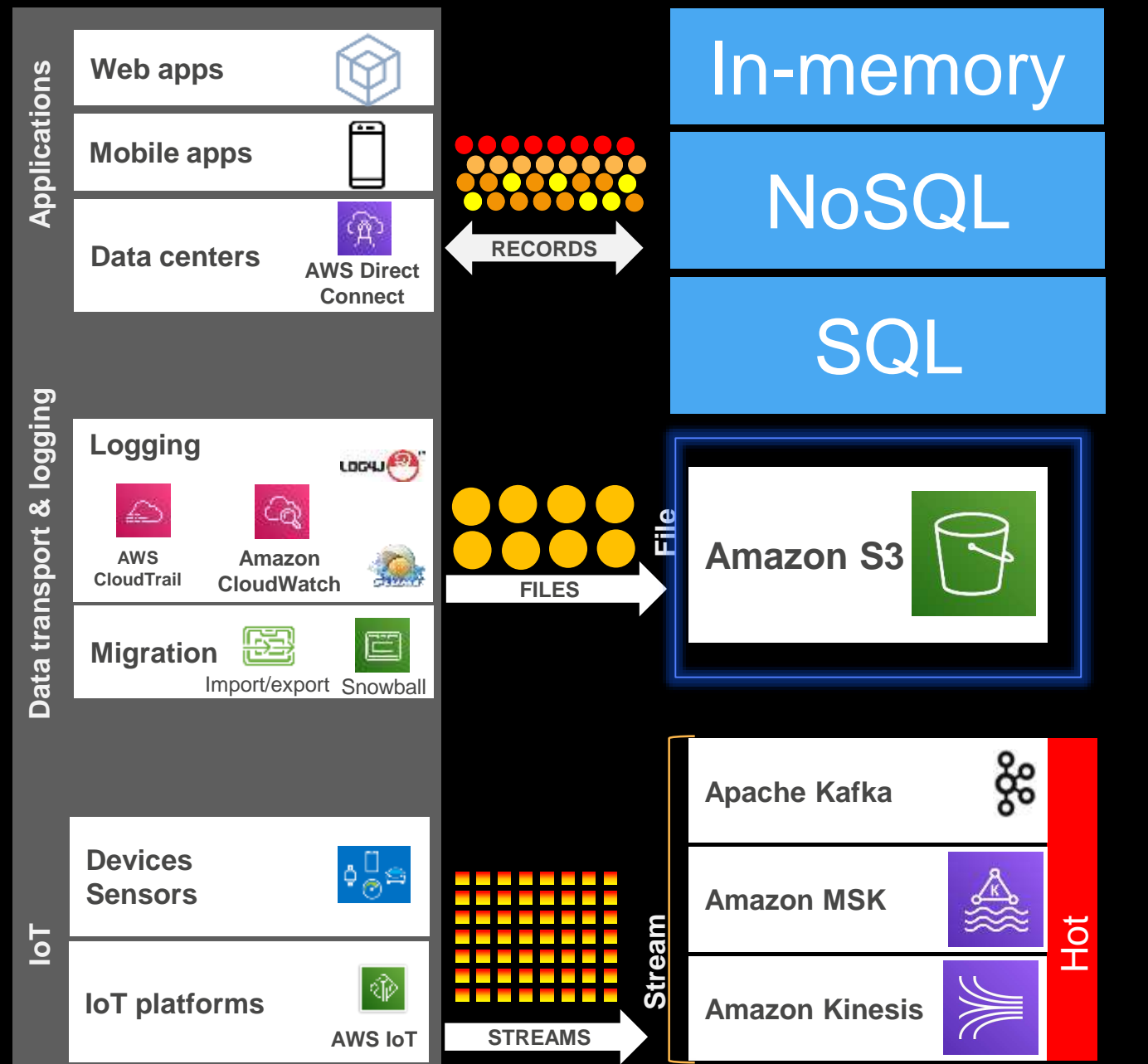
Which stream/message storage should I use?

	Amazon Kinesis Data Streams	Amazon Kinesis Data Firehose	Apache Kafka (on Amazon Elastic Compute Cloud [Amazon EC2])	Amazon Simple Queue Service (Amazon SQS) (Standard)	Amazon SQS (FIFO)
AWS managed	Yes	Yes	No	Yes	Yes
Guaranteed ordering	Yes	No	Yes	No	Yes
Delivery (deduping)	At least once	At least once	At least/At most/exactly once	At least once	Exactly once
Data retention period	7 days	N/A	Configurable	14 days	14 days
Availability	Three AZ	Three AZ	Configurable	Three AZ	Three AZ
Scale/throughput	No limit / ~ shards	No limit / automatic	No limit / ~ nodes	No limits / automatic	300 TPS / queue
Multiple consumers	Yes	No	Yes	No	No
Row/object size	1 MB	Destination row/object size	Configurable	256 KB	256 KB
Cost	Low	Low	Low (+admin)	Low-medium	Low-medium

Collect

Store

File/object storage



Amazon Simple Storage Service (Amazon S3)

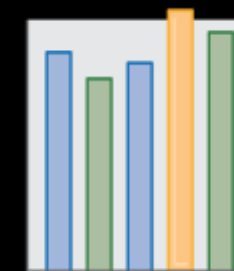
- Managed object storage service built to store and retrieve any amount of data

Use **Amazon S3** as the storage for your data lake

- Natively supported by big data frameworks (Spark, Hive, Presto, and others)
- Decouple storage and compute
 - No need to run compute clusters for storage (unlike HDFS)
 - Can run transient Amazon EMR clusters with Amazon EC2 Spot Instances
 - Multiple & heterogeneous analysis clusters and services can use the same data
- Designed for 99.999999999% durability
- No need to pay for data replication within a region
- Secure – SSL, client/server-side encryption at rest
- Low cost


What about HDFS & data tiering?

- Use **HDFS/local** for working data sets
 - (for example, iterative read on the same data sets)
- Use **Amazon S3 Standard** for frequently accessed data
- Use **Amazon S3 Standard-IA** for less frequently accessed data
- Use **Amazon Glacier** for archiving cold data



S3 analytics

What about metadata?

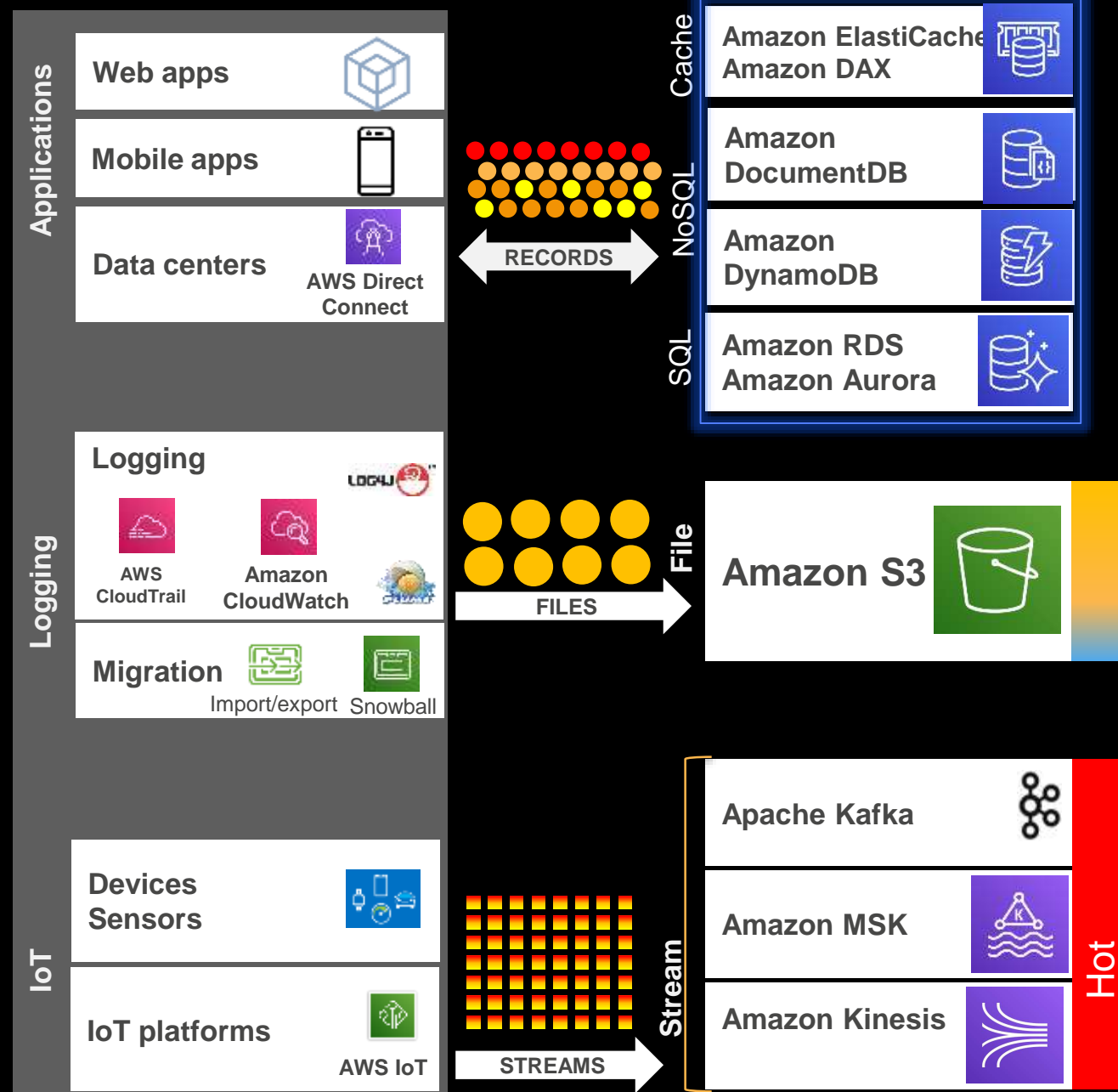
- AWS Glue catalog
 - Hive Metastore compliant
 - Crawlers - Detect new data, schema, partitions
 - Search - Metadata discovery
 - Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum compatible
- Lake Formation - Preview 
- Hive Metastore (Presto, Spark, Hive, Pig)
 - Can be hosted on Amazon RDS



Collect

Store

Cache & database



Amazon ElastiCache

- Managed Memcached or Redis service

Amazon DynamoDB Accelerator (DAX)

- Managed in-memory cache for DynamoDB

Amazon Neptune

- Managed graph DB

Amazon DynamoDB

- Managed key value/document DB

Amazon Relational Database Service (Amazon RDS)

- Managed relational database service

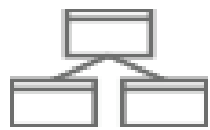
Best practice: Use the right tool for the job



Applications



Data tier



Relational

Referential integrity with strong consistency, transactions, and hardened scale

Complex query support via SQL



Key-value

Low-latency, key-based queries with high throughput and fast data ingestion

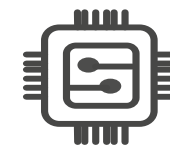
Simple query methods with filters



Document

Indexing and storing of documents with support for query on any property

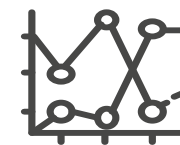
Simple query with filters, projections and aggregates



In-memory

Microsecond latency, key-based queries, specialized data structures

Simple query methods with filters



Graph

Creating and navigating relations between data easily and quickly

Easily express queries in terms of relations

Which data store should I use?

Ask yourself some questions

What is the data structure?

How will the data be accessed?

What is the temperature of the data?

What will the solution cost?

What is the data structure?

Data structure	What to use?
Fixed schema	SQL, NoSQL
Schema-free (JSON)	NoSQL, search
Key/value	In-memory, NoSQL
Graph	GraphDB

How will the data be accessed?

Access patterns	What to use?
Put/get (key, value)	In-memory, NoSQL
Simple relationships → 1:N, M:N	NoSQL
Multi-table joins, transaction, SQL	SQL
Faceting, search	Search
Graph traversal	GraphDB

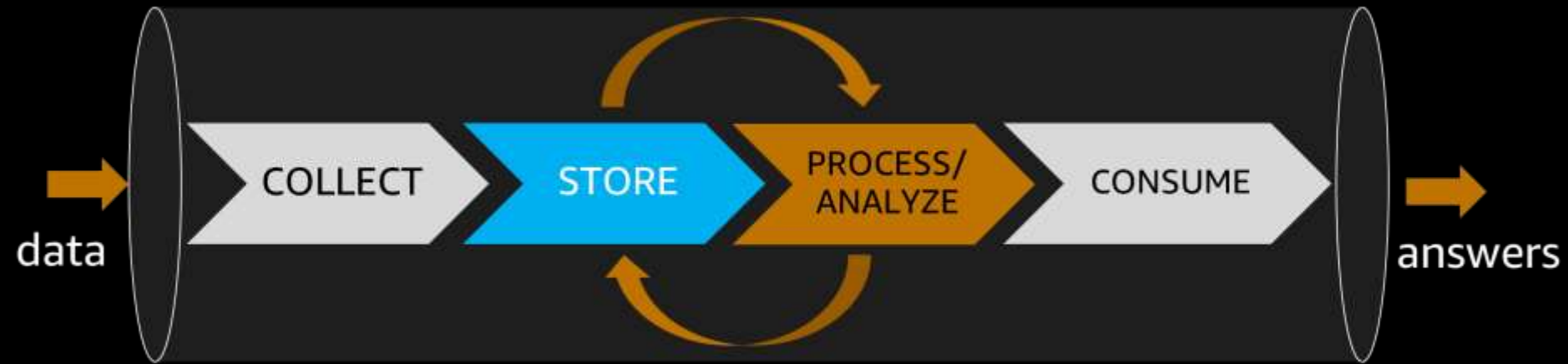
Database characteristics

	Amazon ElastiCache	Amazon DynamoDB + DAX	Amazon Aurora	Amazon RDS	Amazon Elasticsearch Service	Amazon Neptune	Amazon S3 + Amazon Glacier
Use cases	In memory caching	K/V lookups, document store	OLTP, transactional	OLTP, transactional	Log analysis, reverse indexing	Graph	File store
Performance	Ultra high request rate, Ultra low latency	Ultra high request rate, ultra low to low latency	Very high request rate, low latency	High request rate, low latency	Medium request rate, low latency	Medium request rate, low latency	High throughput
Shape	K/V	K/V and document	Relational	Relational	Documents	Node/edges	Files
Size	GB	TB, PB (no limits)	GB, mid TB	GB, low TB	GB, TB	GB, mid TB	GB, TB, PB, EB (no limits)
Cost / GB	\$\$	¢¢ - \$\$	¢¢	¢¢	¢¢	¢¢	¢- ¢4/10
Availability	2 AZ	3 AZ	3 AZ	2 AZ	1-2 AZ	3 AZ	3 AZ
VPC support	Inside VPC	VPC endpoint	Inside VPC	Inside VPC	Outside or inside VPC	Inside VPC	VPC endpoint

Hot data

Warm data

Cold data



Process/
analyze

Interactive & batch analytics

Process/analyze

Amazon Elasticsearch Service

Managed service for Elasticsearch

Amazon Redshift & Amazon Redshift Spectrum

Managed data warehouse

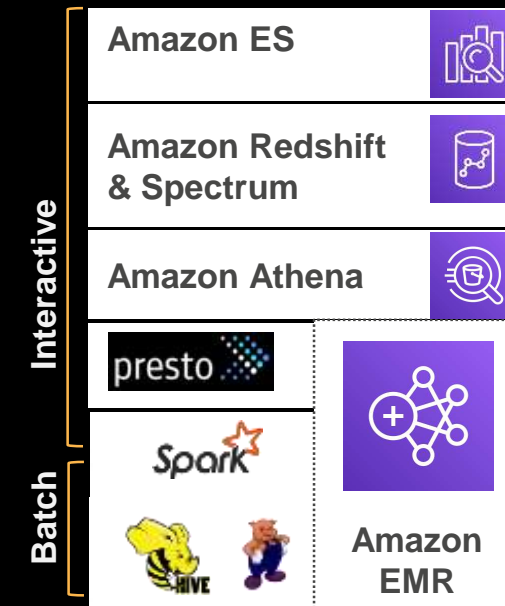
Spectrum enables querying S3

Amazon Athena

Serverless interactive query service

Amazon EMR

Managed Hadoop framework for running Apache Spark, Flink, Presto, Tez, Hive, Pig, HBase, and others



Stream/real-time analytics

Process/analyze

Spark Streaming on Amazon EMR

Amazon Kinesis Data Analytics

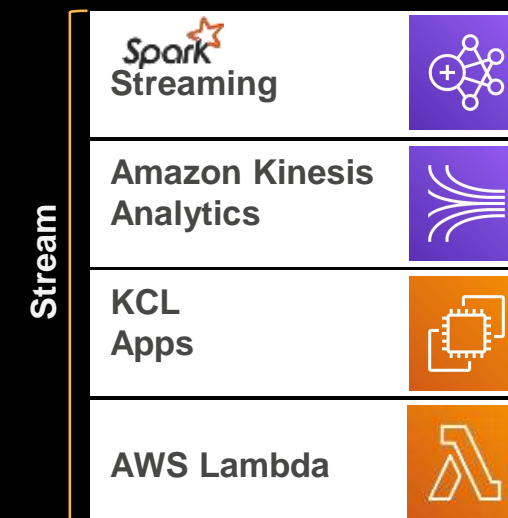
- Managed service for running SQL on streaming data

Amazon KCL

- Amazon Kinesis Client Library

AWS Lambda

- Run code serverless (without provisioning or managing servers)
- Services such as S3 can publish events to AWS Lambda
- AWS Lambda can pool event from a Kinesis

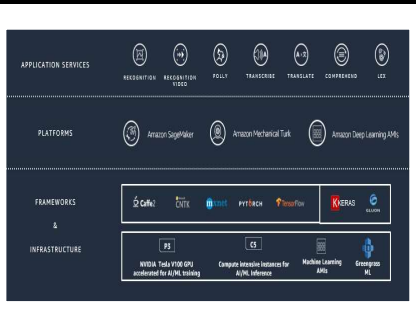


Predictive analytics

Process/analyze

Predictive

Amazon ML



APPLICATION SERVICES



Developers

PLATFORMS



Data scientists
Deep learning
experts

FRAMEWORKS



&

INFRASTRUCTURE



Which analytics should I use?

Batch

Takes minutes to hours

Example: Daily/weekly/monthly reports

Amazon EMR (MapReduce, Hive, Pig, Spark)

Interactive

Takes seconds

Example: Self-service dashboards

Amazon Redshift, Amazon Athena, Amazon EMR (Presto, Spark)

Stream

Takes milliseconds to seconds

Example: Fraud alerts, one-minute metrics

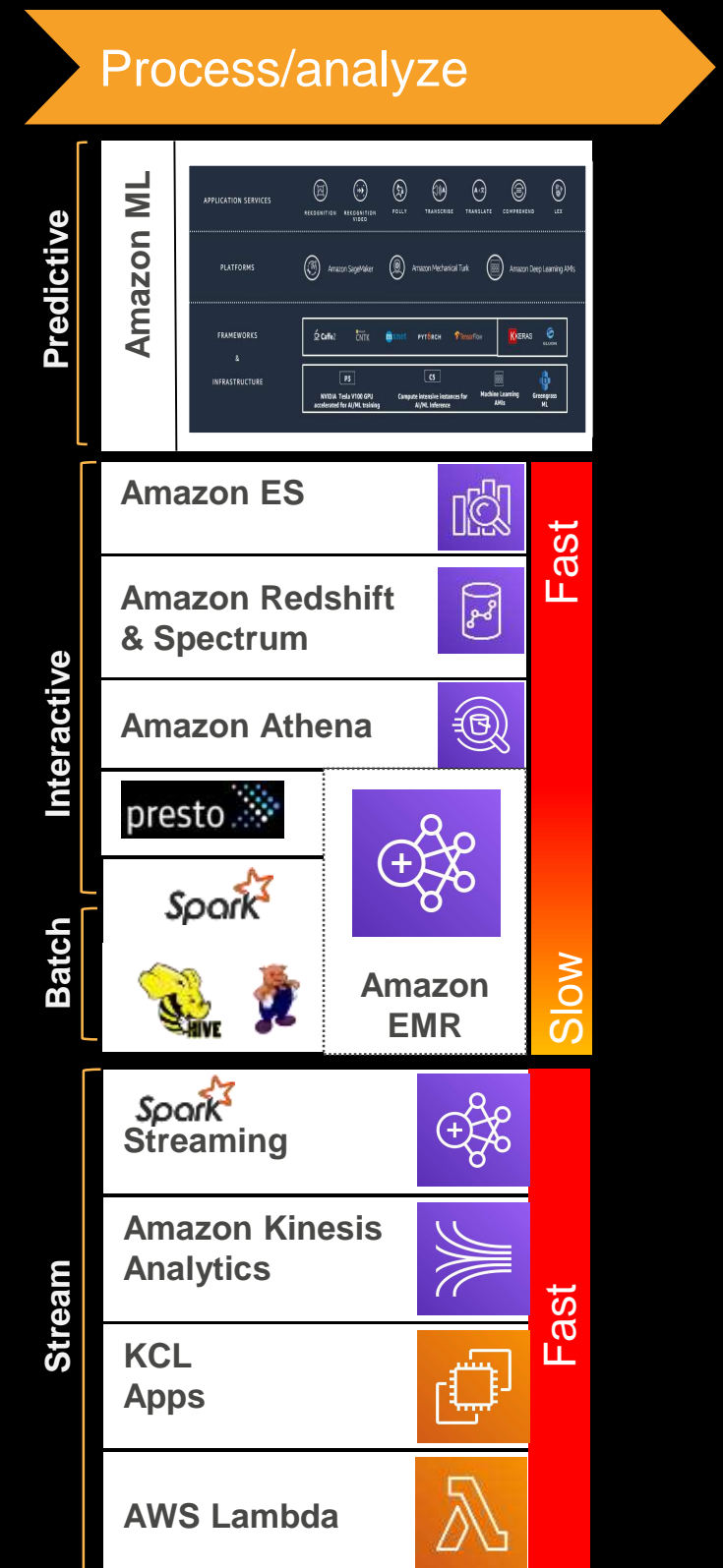
Amazon EMR (Spark Streaming), Amazon Kinesis Data Analytics, Amazon KCL, AWS Lambda, and others

Predictive

Takes milliseconds (real-time) to minutes (batch)

Example: Fraud detection, forecasting demand, speech recognition

Amazon SageMaker, Amazon Polly, Amazon Rekognition, Amazon Transcribe, Amazon Translate, Amazon EMR (Spark ML), Amazon Deep Learning AMI (MXNet, TensorFlow, Theano, Torch, CNTK, and Caffe)



Which stream processing technology should I use?

	Amazon EMR (Spark Streaming)	KCL application	Amazon Kinesis analytics	AWS Lambda
Managed service	Yes	No (EC2 + Auto Scaling)	Yes	Yes
Serverless	No	No	Yes	Yes
Scale / throughput	No limits / ~ nodes	No limits / ~ nodes	No Limits / automatic	No limits / automatic
Availability	Single AZ	Multi-AZ	Multi-AZ	Multi-AZ
Programming languages	Java, Python, Scala	Java, others through MultiLangDaemon	ANSI SQL with extensions	Node.js, Java, Python, .Net Core
Sliding window functions	Build-in	App needs to implement	Built-in	No
Reliability	KCL and Spark checkpoints	Managed by Amazon KCL	Managed by Amazon Kinesis Data Analytics	Managed by AWS Lambda

	Amazon Redshift	Amazon Redshift Spectrum	Amazon Athena	Amazon EMR		
				Presto	Spark	Hive
Use case	Optimized for data warehousing	Query S3 data from Amazon Redshift	Interactive queries over S3 data	Interactive query	General purpose	Batch
Scale/throughput	~Nodes	~Nodes	Automatic	~ Nodes		
Managed service	Yes	Yes	Yes, Serverless	Yes		
Storage	Local storage	Amazon S3	Amazon S3	Amazon S3, HDFS		
Optimization	Columnar storage, data compression, and zone maps	AVRO, PARQUET TEXT, SEQ RCFILE, ORC, etc.	AVRO, PARQUET TEXT, SEQ RCFILE, ORC, etc.	Framework dependent		
Metadata	Amazon Redshift catalog	AWS Glue catalog	AWS Glue catalog	AWS Glue catalog or Hive Meta-store		
Auth/access controls	IAM, users, groups, and access controls	IAM, users, groups, and access controls	IAM	IAM, LDAP, & Kerberos		
UDF support	Yes (Scalar)	Yes (Scalar)	No	Yes		

Fastest

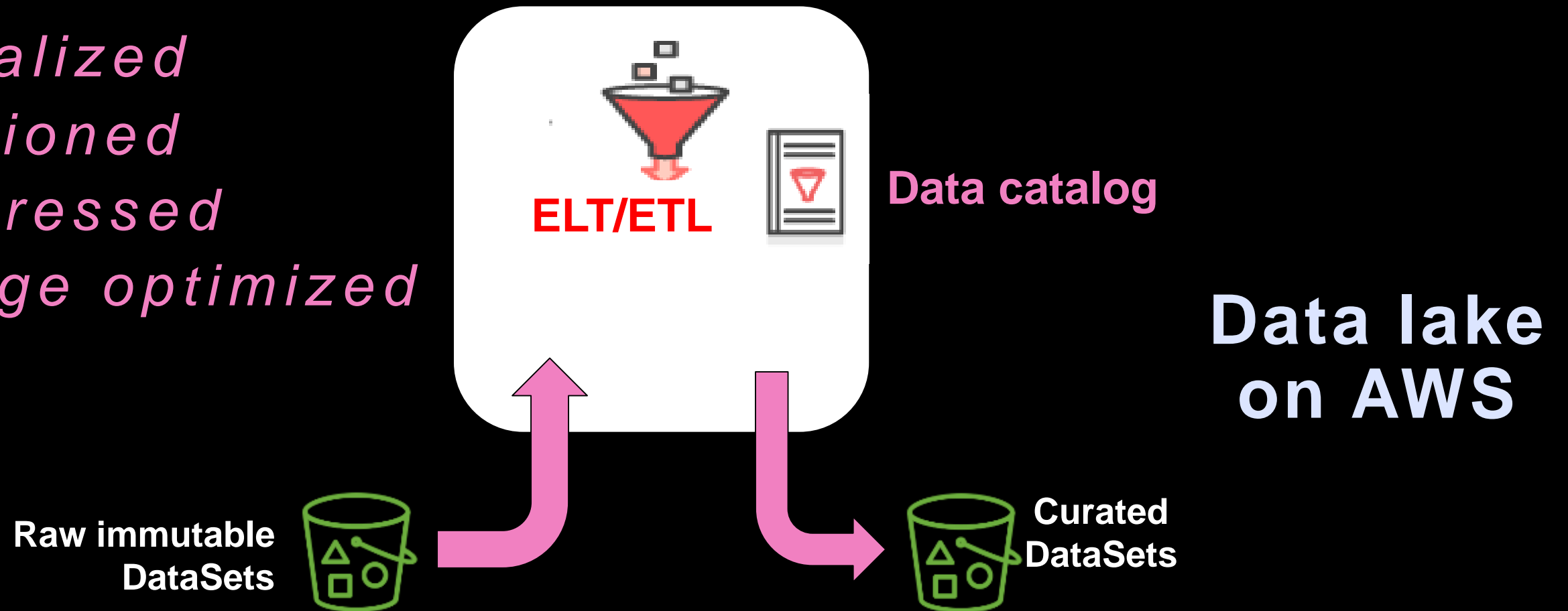
Slow

ELT/ETL: Preparing raw data for consumption

Raw data stored in data lake

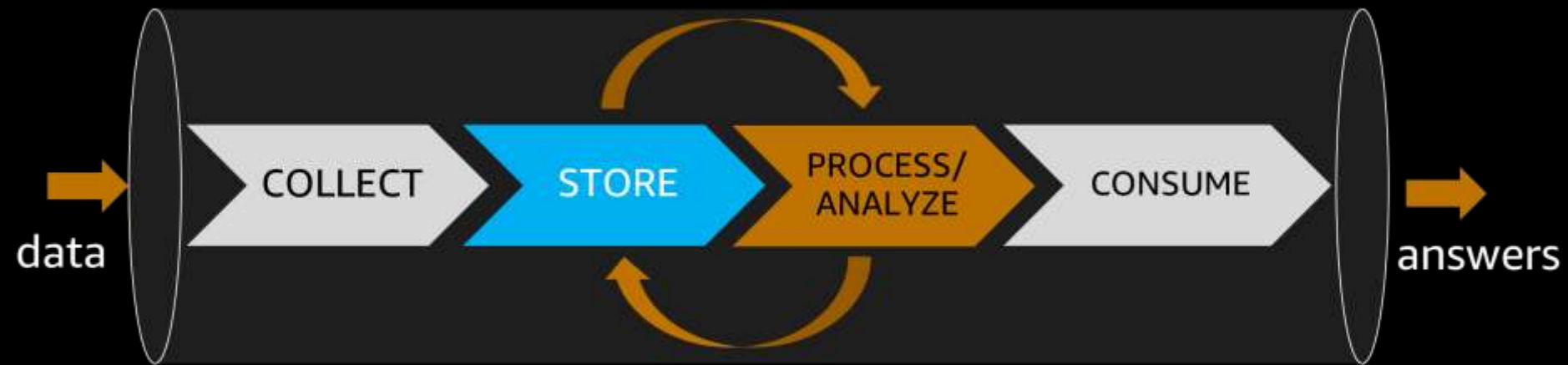
Preparation

*Normalized
Partitioned
Compressed
Storage optimized*

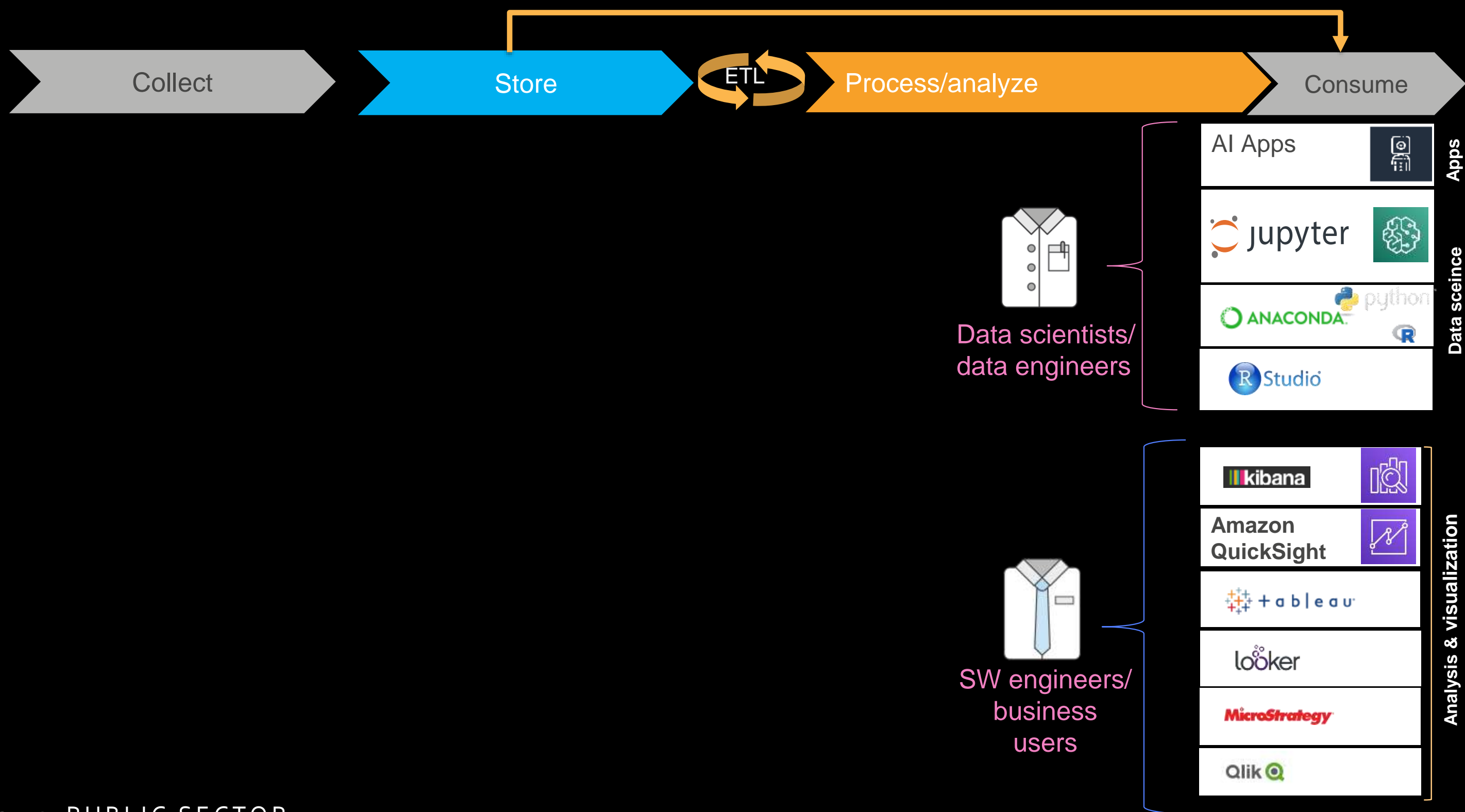


Which ELT/ETL tool should I use?

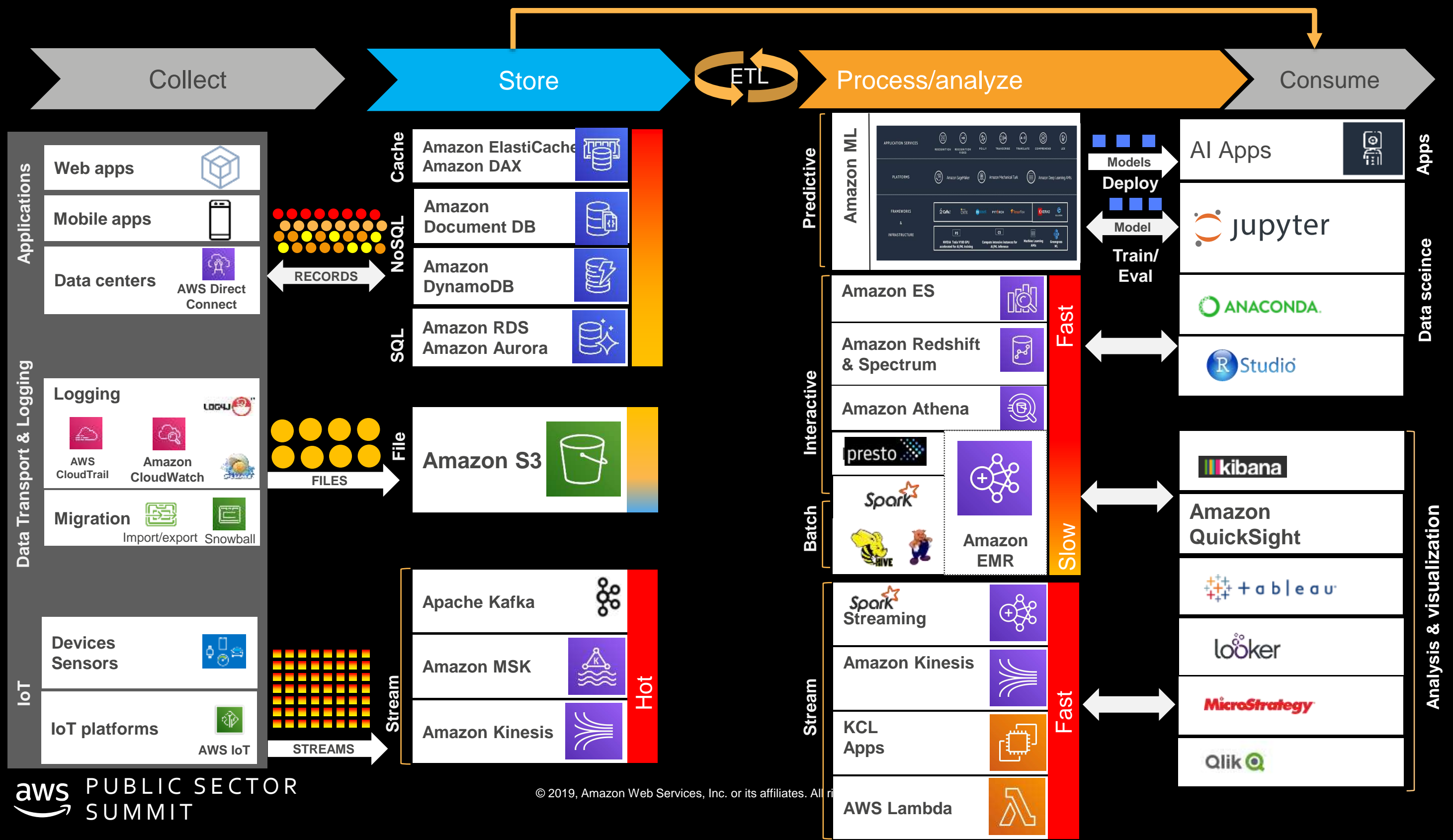
	AWS Glue ETL	AWS Data Pipeline	Amazon Data Migration Service (Amazon DMS)	Amazon EMR	Apache NiFi	Partner solution
Use case	Serverless ETL	Data workflow service	Migrate databases (and to/from data lake)	Customize developed Hadoop/Spark ETL	Automate the flow of data between systems	Rich partner ecosystem for ETL
Scale/throughput	~DPUs	~Nodes through EMR cluster	EC2 instance type	~Nodes	Self managed	Self manager or through partner
Managed service	Clusterless	Managed	Managed EC2 on your behalf	Managed EC2 on your behalf	Self managed on Amazon EMR or Marketplace	Self manager or through partner
Data sources	S3, RDBMS, Amazon RedShift, DynamoDB	S3, JDBC, DynamoDB, custom	RDBMS, data warehouses, Amazon S3* (*limited)	Various Hadoop/Spark managed	Various through rich processor framework	Various
Skills needed	Wizard for simple mapping, code snippets for advanced ETL	Wizard and code snippets	Wizard and drag/drop	Hadoop/Spark coding	NiFi processors and some coding	Self manager or through partner



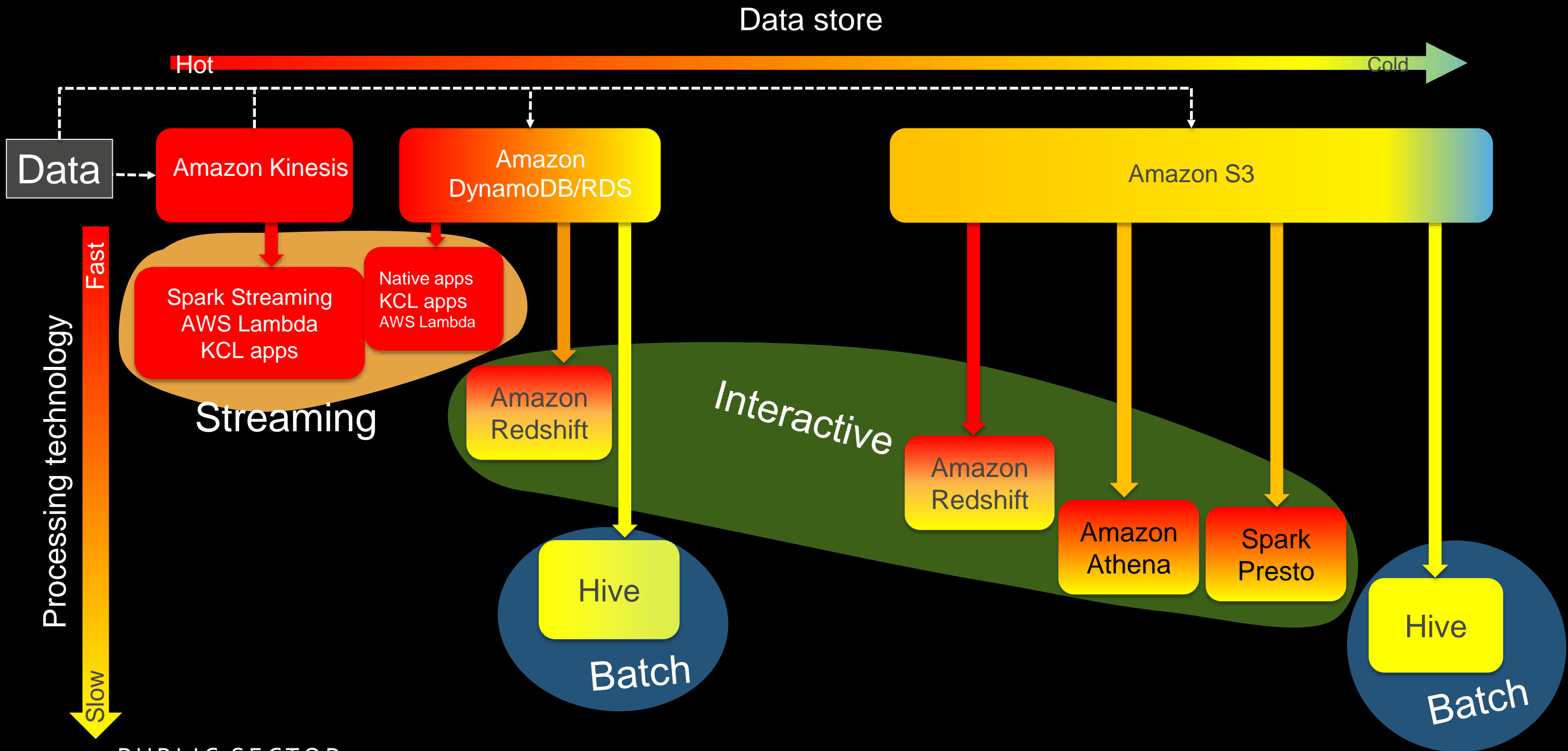
Consume



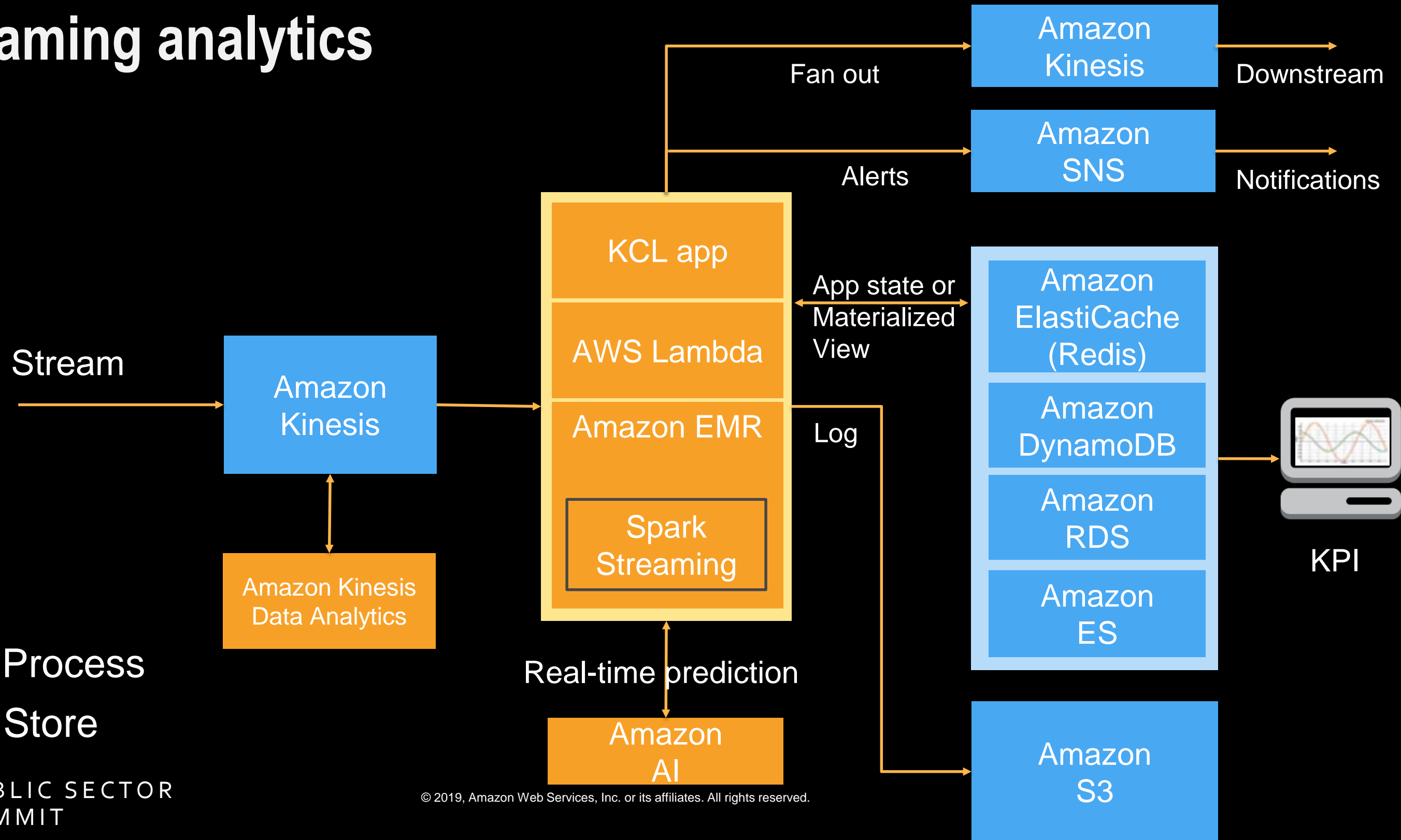
Putting it all together



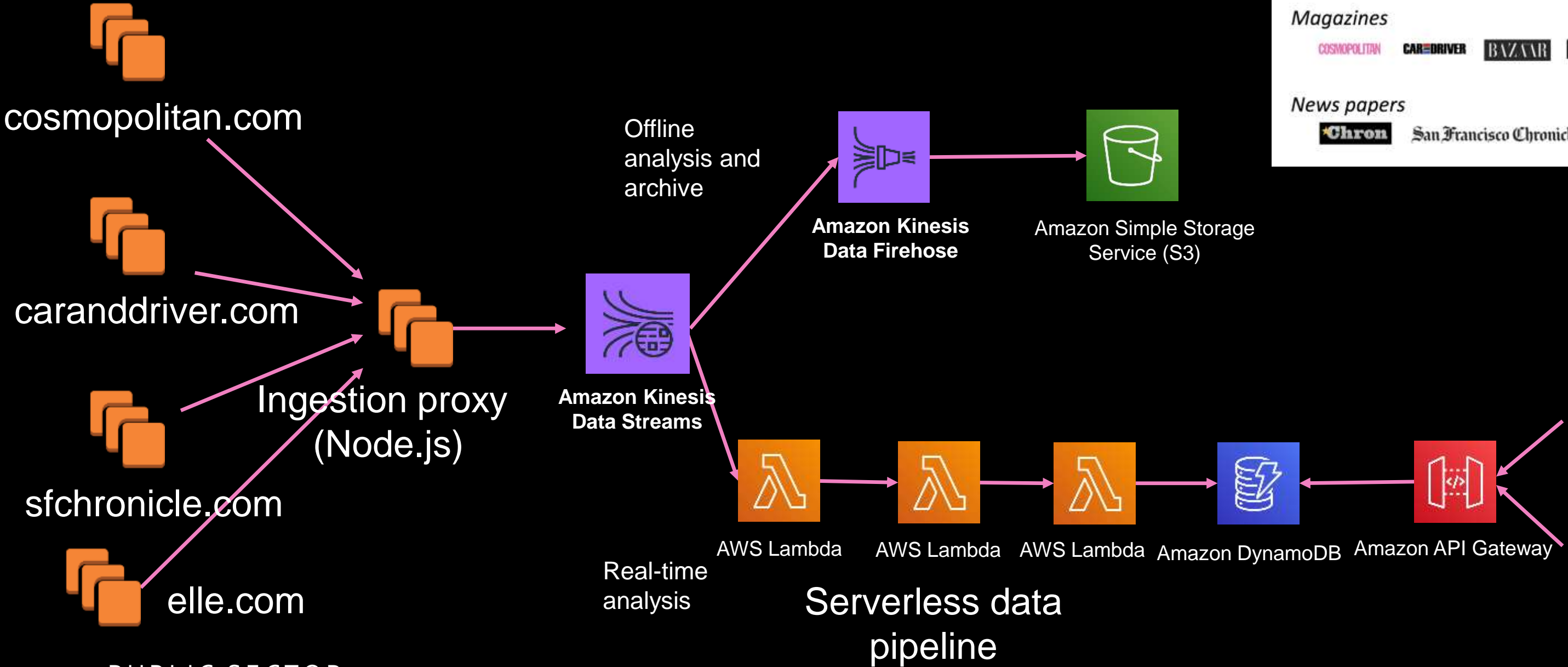
Design patterns



Streaming analytics



Hearst's serverless data pipeline



HEARST
Media and information company

Television

Magazines

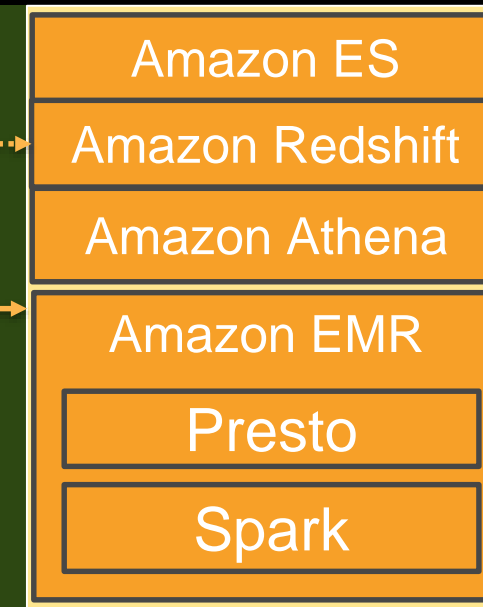
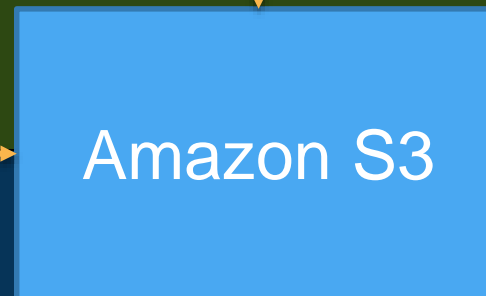
News papers

Interactive & batch analytics

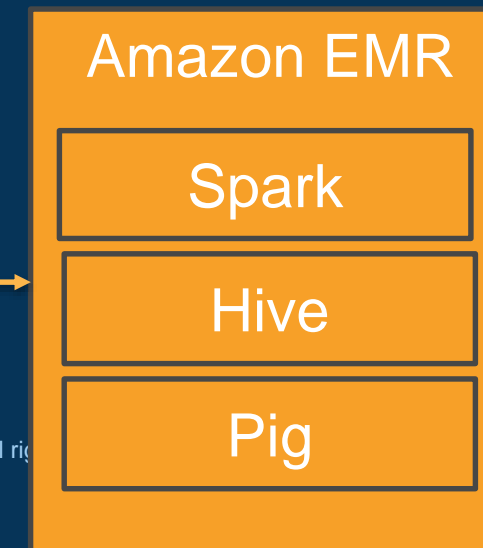
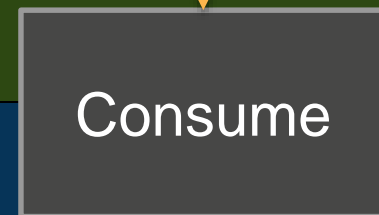
Stream



Files



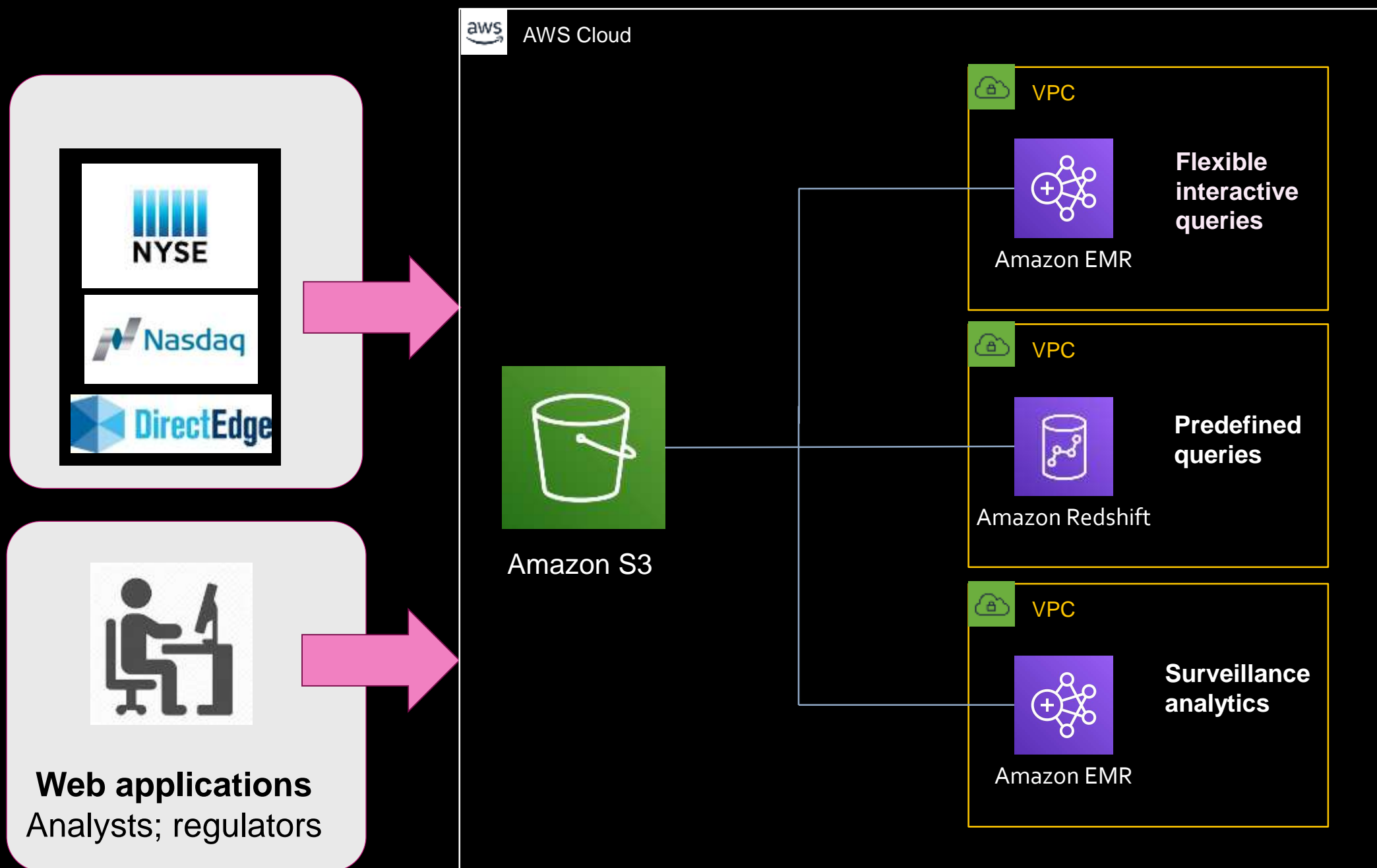
Interactive



Batch



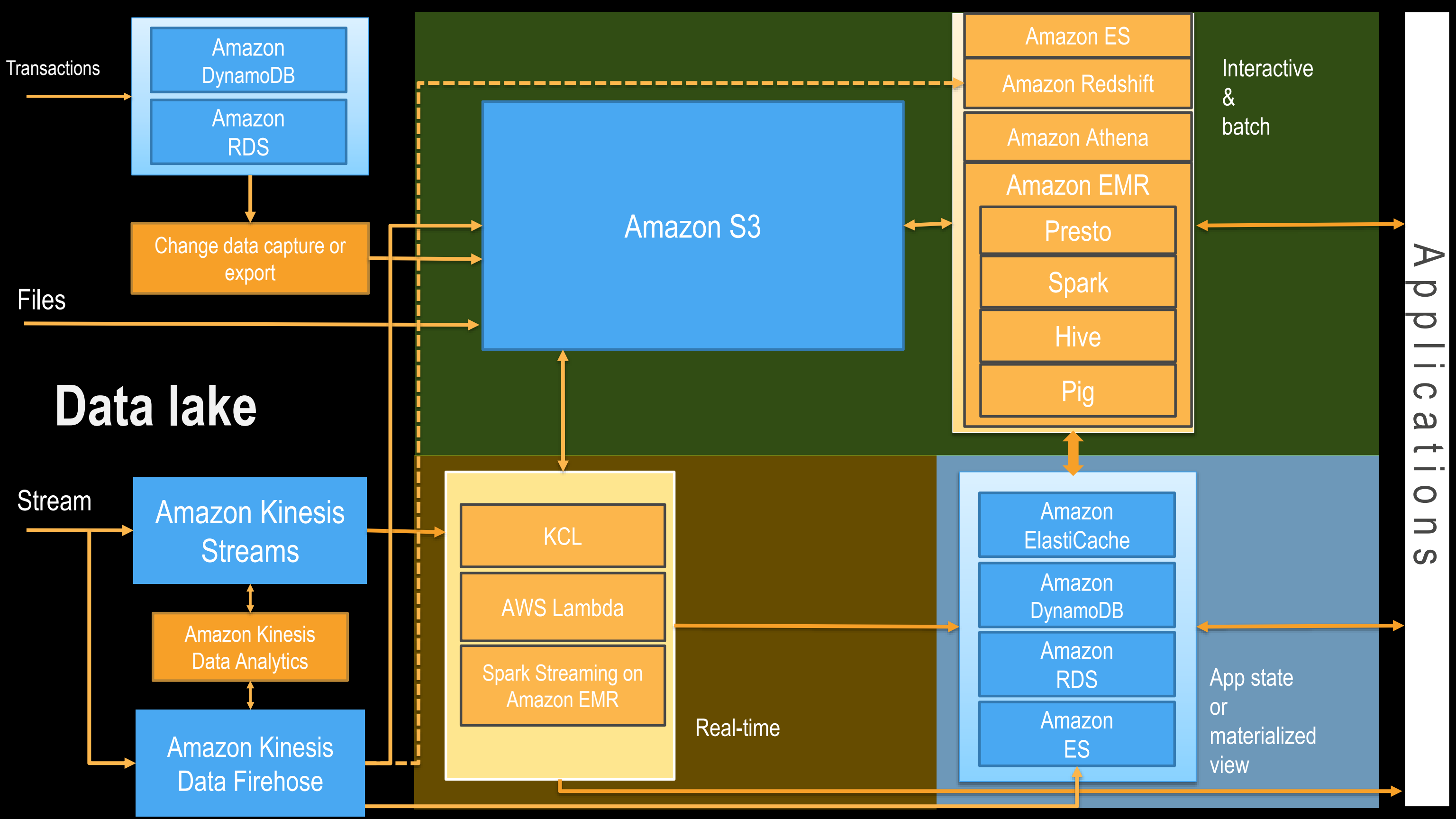
FINRA: Migrating to AWS



Petabytes of data generated on-premises, brought to AWS, and stored in S3

Thousands of analytical queries performed on Amazon EMR and Amazon Redshift

Stringent security requirements met by leveraging VPC, VPN, encryption at-rest and in-transit, AWS CloudTrail, and database auditing



Summary

Build decoupled systems

- Data → Store → Process → Store → Analyze → Answers

Use the right tool for the job

- Data structure, latency, throughput, access patterns

Leverage AWS managed and serverless services

- Scalable/elastic, available, reliable, secure, no/low admin

Use log-centric design patterns

- Immutable logs, data lake, materialized views

Be cost-conscious

- Big data ≠ big cost

Thank you!

Manos Samatas
samatas@amazon.com



Please complete the
session survey.