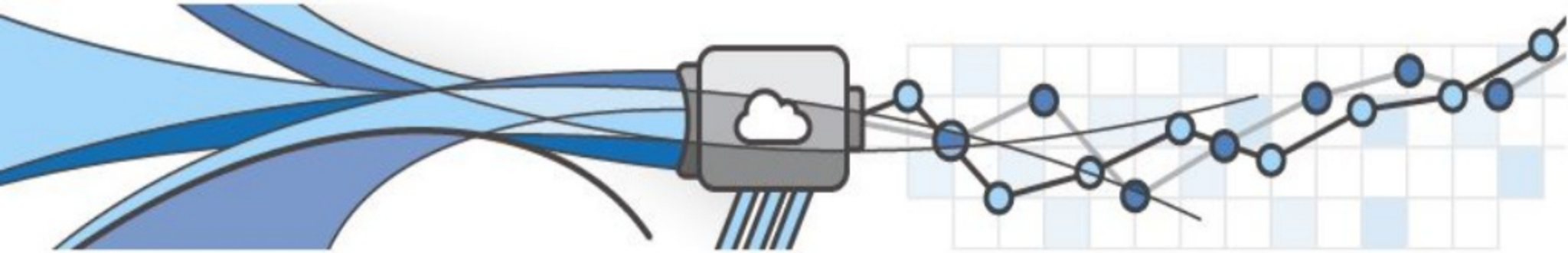


Big Data Solutions Day
Introductory Session

Wesley Wilk
Solutions Architect, AWS

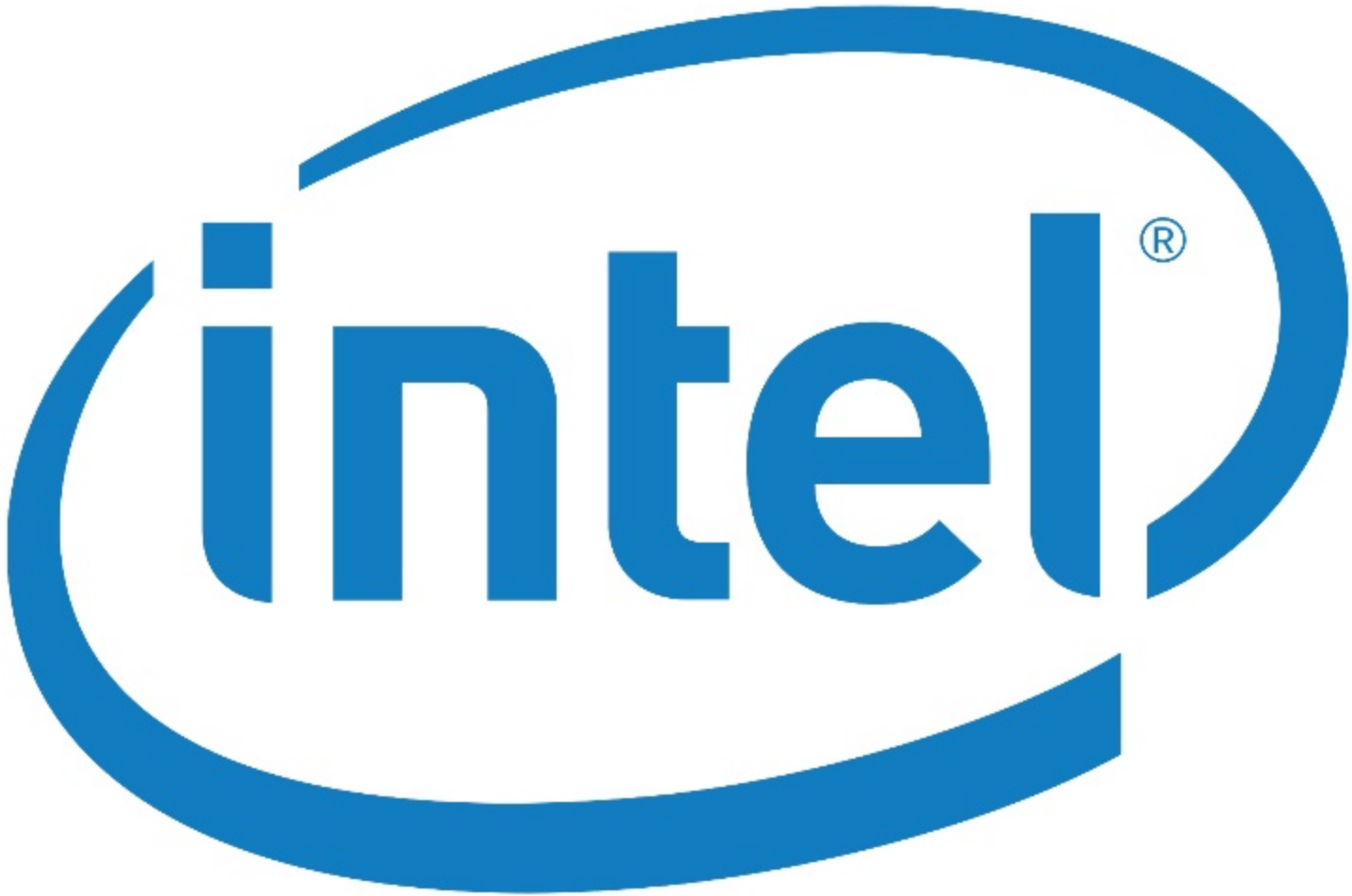
intel

amazon web services

Today

12:30 PM	Introductory Session
2:00 PM	Best Practices on Real-time Streaming Analytics
3:00 PM	Break
3:15 PM	Getting started with Amazon Machine Learning
4:15 PM	Building your first big data application on AWS

Thank you to our Sponsor



amazon web services

AWS In 2016:

64%

YOY Growth
Q1 2015 TO Q1 2016

1,000,000+

Active Customers Per Month

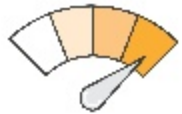
10x

Compute Capacity In Use

$10B+

amazon web services

Largest Ecosystem of ISVs



Solutions vetted by the AWS Partner Competency Program

Data Integration
Move, synchronize, cleanse, and manage data

Data Analysis & Visualization
Turn data into actionable insight and enhance decision making

Infrastructure Intelligence
Harness data generated from your systems and infrastructure

Advanced Analytics
Anticipate future behaviors and conduct what-if analysis

alooma
alteryx
ATTUNITY
byte
informatica
Matillion
snapLogic
talend

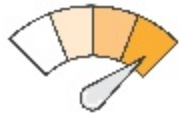
CHARTIO
looker
MicroStrategy
tableau
TIBCO Jaspersoft

splunk>
sumologic

CIVIS ANALYTICS
databricks
MAPR
Qubole
SAP
snowflake
TREASURE DATA

amazon web services

Largest Ecosystem of Integrators



Qualified consulting partners with AWS big data expertise

North America

2nd Watch
47 Lining
accenture — High performance. Delivered.
clearscale
Cloudwick
Cognizant
CORECOMPETE
CSC
DAY1 SOLUTIONS
FULL 360
MarketShare
NorthBay
PARIVEDA SOLUTIONS
RELUS
slalom
softserve
TriNimbus

Europe, Middle East and Africa

BEEVA
comSysto
Globant
KCOM
Reply — storm
tecRACER

Asia Pacific and Japan

blazeclan — Cloud IT Better
classmethod
cloudpack
Minjar
NRI
TO THE NEW DIGITAL

amazon web services

Big Data Analytics in AWS Marketplace

Thousands of products pre-integrated with the AWS Cloud.
290 specific to big data

- Deploy when you need it, 1-Click launch in multiple regions around the world
- Solutions are pre-configured and ready to run on AWS
- Metered pricing by the hour. Pay only for what you use. Volume licensing available

Analysis & Visualization

Extract valuable information from your historical and current data

tableau software
TIBCO Jaspersoft
SAP
MicroStrategy

Advanced Analytics

Predict future business performance; location, text, social, sentiment analysis

TIBCO Spotfire
ZEMENTIS Empowering Big Data Insights
bigml
MAPLARGE we bring DATA to life

Data Integration

Extract, migrate, or prepare and clean your data for accurate analysis

ATTUNITY
Matillion
informatica
syncsort

http://aws.analytic.business

amazon web services

Proven Customer Success

The vast majority of big data use cases deployed in the cloud today run on AWS.

amazon web services

Big Data is Breaking Traditional IT Infrastructure

- Too many tools to setup, manage & scale
- Unsustainable costs
- Difficult & undifferentiated heavy lifting just to get started
- New & expensive skills
- Bigger responsibility (sensitive data)

amazon web services

The Evolution of Big Data Processing

Descriptive

What happened before

Dashboards; Traditional query & reporting

Real-time

What's happening now

Clickstream analysis; Ad bidding; streaming data

Predictive

Probability of "x" happening

Inventory forecasting; Fraud detection; Recommendation engines

Big Data on AWS

1. Broadest & Deepest Functionality
2. Computational Power that is Second to None
3. Petabyte-scale Analytics Made Affordable
4. Easiest & Most Powerful Tools for Migrating Data
5. Security You Can Trust

1. Broadest & Deepest Functionality

Big Data Storage

Data Warehousing

Real-time Streaming

Distributed Analytics (Hadoop, Spark, Presto)

NoSQL Databases

Business Intelligence

Relational Databases

Internet of Things (IoT)

Machine Learning

Elasticsearch

Server-less Compute

amazon web services

Core Components for Big Data Workloads

Big Data Storage for Virtually All AWS Services

Amazon S3

Store anything

Object storage

Scalable

99.999999999% durability

Extremely low cost

amazon web services

Real-time Streaming Data

Amazon
Kinesis

Real-time processing

High throughput; elastic

Easy to use

Integration with S3, EMR, Redshift, DynamoDB

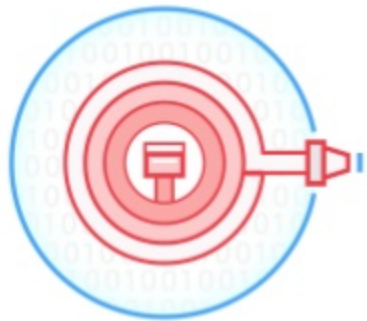
amazon web services

Amazon Kinesis: Streaming Data Made Easy

Services make it easy to capture, deliver and process streams on AWS

Amazon Kinesis Streams

- For Technical Developers
- Build your own custom applications that process or analyze streaming data



Amazon Kinesis Firehose

- For all developers, data scientists
- Easily load massive volumes of streaming data into S3, Amazon Redshift and Amazon Elasticsearch

Amazon Kinesis Analytics

- For all developers, data scientists
- Easily analyze data streams using standard SQL queries
- **Coming soon**

amazon web services

Distributed Analytics

Amazon EMR

Managed Hadoop framework

Spark, Presto, Hbase, Tez, Hive, etc.

Cost effective; Easy to use

On-demand and spot pricing

HDFS & S3 File systems

amazon web services

Fast & Flexible NoSQL Database Service

Amazon
DynamoDB

NoSQL Database

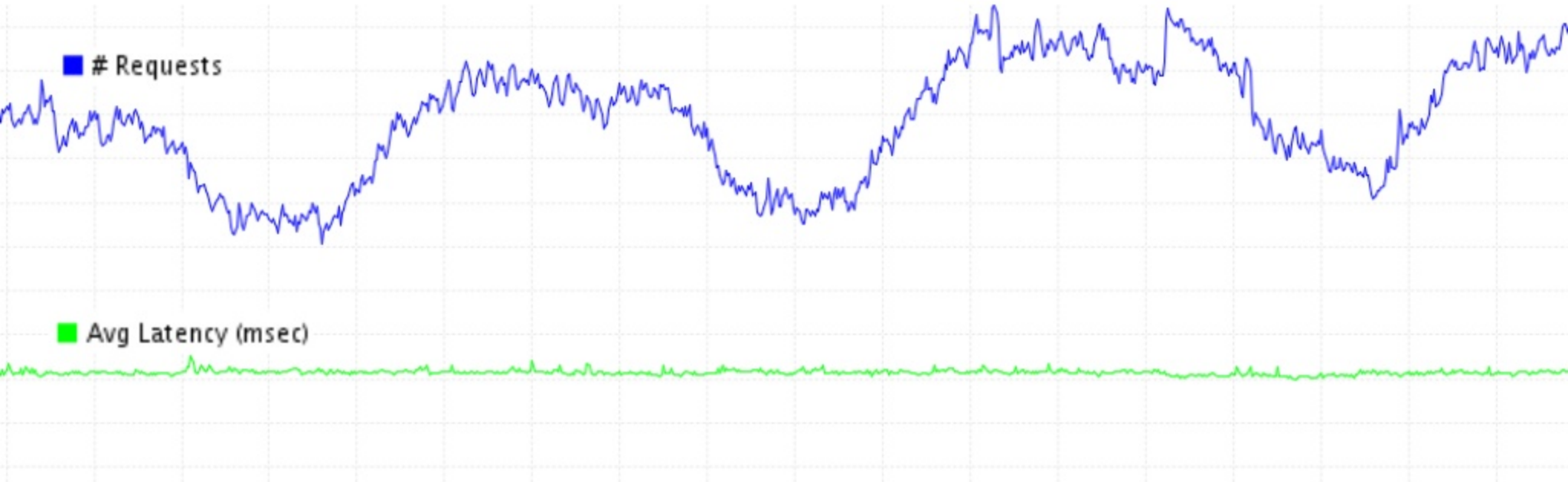
Seamless scalability

Zero admin

Single digit millisecond latency

amazon web services

Durable Low Latency – At Scale



WRITES
Continuously replicated to 3 AZ's
Quorum acknowledgment
Persisted to disk (custom SSD)

READS
Strongly or eventually consistent
No trade-off in latency

amazon web services

Fully Managed Petabyte-scale Data Warehouse

Amazon
Redshift

Relational data warehouse

Massively parallel; Petabyte scale

Fully managed

HDD and SSD Platforms

$1,000/TB/Year; start at $0.25/hour

amazon web services

Scale from 160GB to 2PB
Push-button Global DR
Built in end-to-end security

Amazon Redshift has security built-in

SSL to secure data in transit

Encryption to secure data at rest

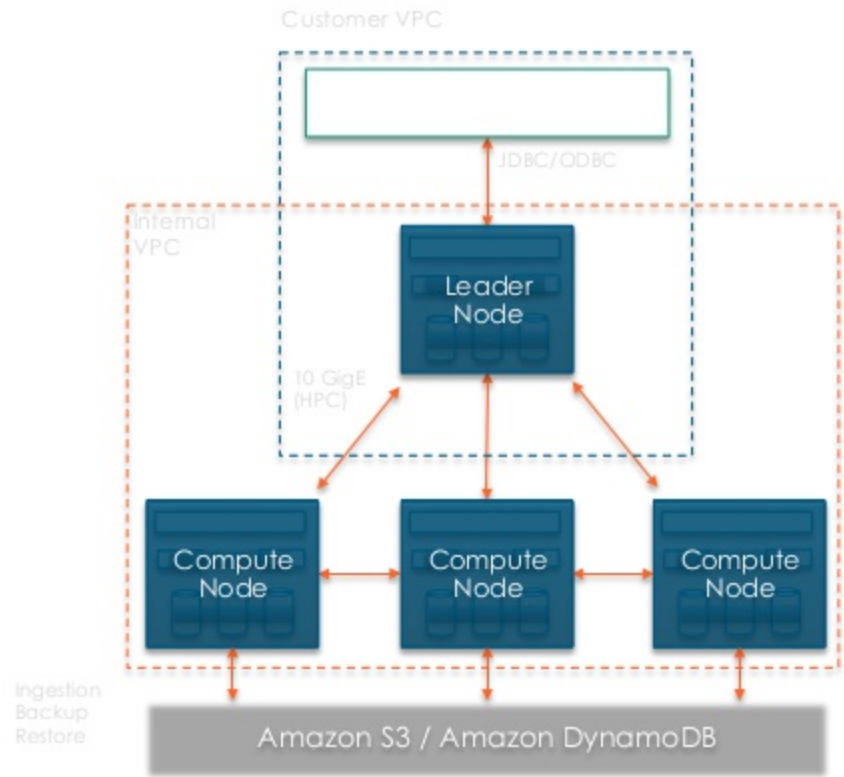
- AES-256; hardware accelerated
- All blocks on disks and in Amazon S3 encrypted
- HSM Support

No direct access to compute nodes

Audit logging, AWS CloudTrail, AWS KMS integration

Amazon VPC support

SOC 1/2/3, PCI-DSS Level 1, FedRAMP, HIPAA



amazon web services

Machine Learning

Amazon
Machine Learning

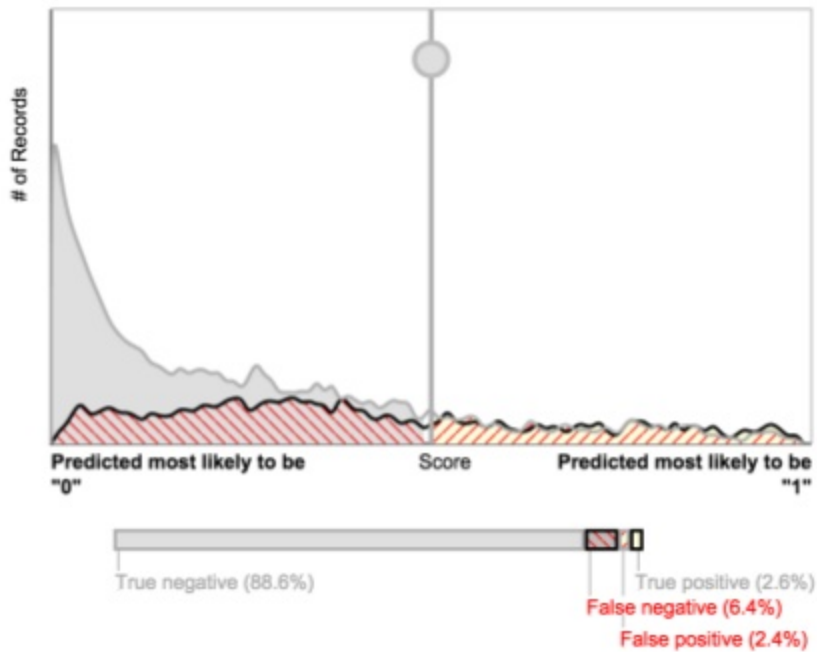
Easy to use, managed machine learning service built for developers

Robust, powerful machine learning technology based on Amazon's internal systems

Create models using your data already stored in the AWS cloud

Deploy models to production in seconds

Simple Model Tuning



Trade-off based on score threshold 0.5

Reset score threshold (0.5)

- **91% are correct**
  607 true positive
  20,933 true negative
- **9% are errors**
  567 false positive
  1,507 false negative

- 5% of the records are predicted as "1"
- 95% of the records are predicted as "0"

Save score threshold at 0.50

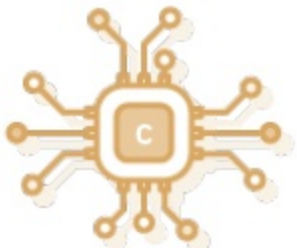
› Advance metrics

Sample Use Cases

Fraud detection	Detecting fraudulent transactions, filtering spam emails, flagging suspicious reviews, …
Personalization	Recommending content, predictive content loading, improving user experience, …
Targeted marketing	Matching customers and offers, choosing marketing campaigns, cross-selling and up-selling, …
Content classification	Categorizing documents, matching hiring managers and resumes, …
Churn prediction	Finding customers who are likely to stop using the service, free-tier upgrade targeting, …
Customer support	Predictive routing of customer emails, social media listening, …

2. Computational Power that is Second to None

Over 2x computational instance types as any other cloud platform to address a wide range of big data use cases.



Compute Optimized (C3, C4)

Memory Optimized (R3, X1)

Storage Optimized (I2, D2)

GPU Instances (G2)

Plus general purpose (M4), low cost burstable (T2), and dedicated instances

amazon web services

Intel® Processor Technologies

Intel® AVX – Dramatically increases performance for highly parallel HPC workloads such as life science engineering, data mining, financial analysis, media processing

Intel® AES-NI – Enhances security with new encryption instructions that reduce the performance penalty associated with encrypting/decrypting data

Intel® Turbo Boost Technology – Increases computing power with performance that adapts to spikes in workloads

Intel Transactional Synchronization (TSX) Extensions – Enables execution of transactions that are independent to accelerate throughput

P state & C state control – provides granular performance tuning for cores and sleep states to improve overall application performance

3. Affordable Petabyte-scale Analytics

AWS helps customers maximize the value of Big Data investments while reducing overall IT costs

Amazon Glacier

$7.16 / TB / month

Data Archiving

Amazon S3

$28.16 / TB / month

Secure, Highly Durable storage

Amazon Kinesis

$0.035 / GB

Real-time streaming data load

Amazon EMR

$0.15 / hr

10-node Spark Cluster

Amazon Redshift

$0.25 / hr

Petabyte-scale Data Warehouse

amazon web services

Volume

Avg 5.5B rows per day.
Peak of 14B rows

Performance

Queries running faster.
Loads finish before 11PM

Scale

Loading more data than ever stored by legacy DWH

NASDAQ migrated to Amazon Redshift, achieving cost savings of 57% compared to legacy system

amazon web services

4. Easy, Powerful Tools for Migrating Data

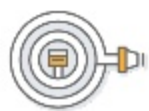
AWS provides the broadest range of tools for easy, fast, secure data movement to and from the AWS cloud.

AWS Direct Connect

AWS Snowball

Database Migration Service

ISV Connectors



Amazon Kinesis Firehose

Amazon S3 Transfer Acceleration

AWS Storage Gateway

amazon web services

What is Snowball? Petabyte scale data transport

Ruggedized
case
"8.5G Impact"

E-ink shipping
label

50TB & 80TB
10G network

Rain & dust
resistant

Tamper-resistant
case & electronics

All data encrypted
end-to-end

amazon
web services

AWS Database Migration Service

Start your first migration in 10 minutes or less

Keep your apps running during the migration

Replicate within, to, or from Amazon EC2 or Amazon RDS

Move data to the same or different database engine

amazon web services

5. Security & Compliance You Can Trust

- AWS manages 1800+ security controls **so you don't have to**
- You benefit from an environment built for the most security sensitive organizations
- You get to define the right security controls for your workload sensitivity
- You always have full ownership and control of your data

Deter

misconduct by enforcing the rules

Detect

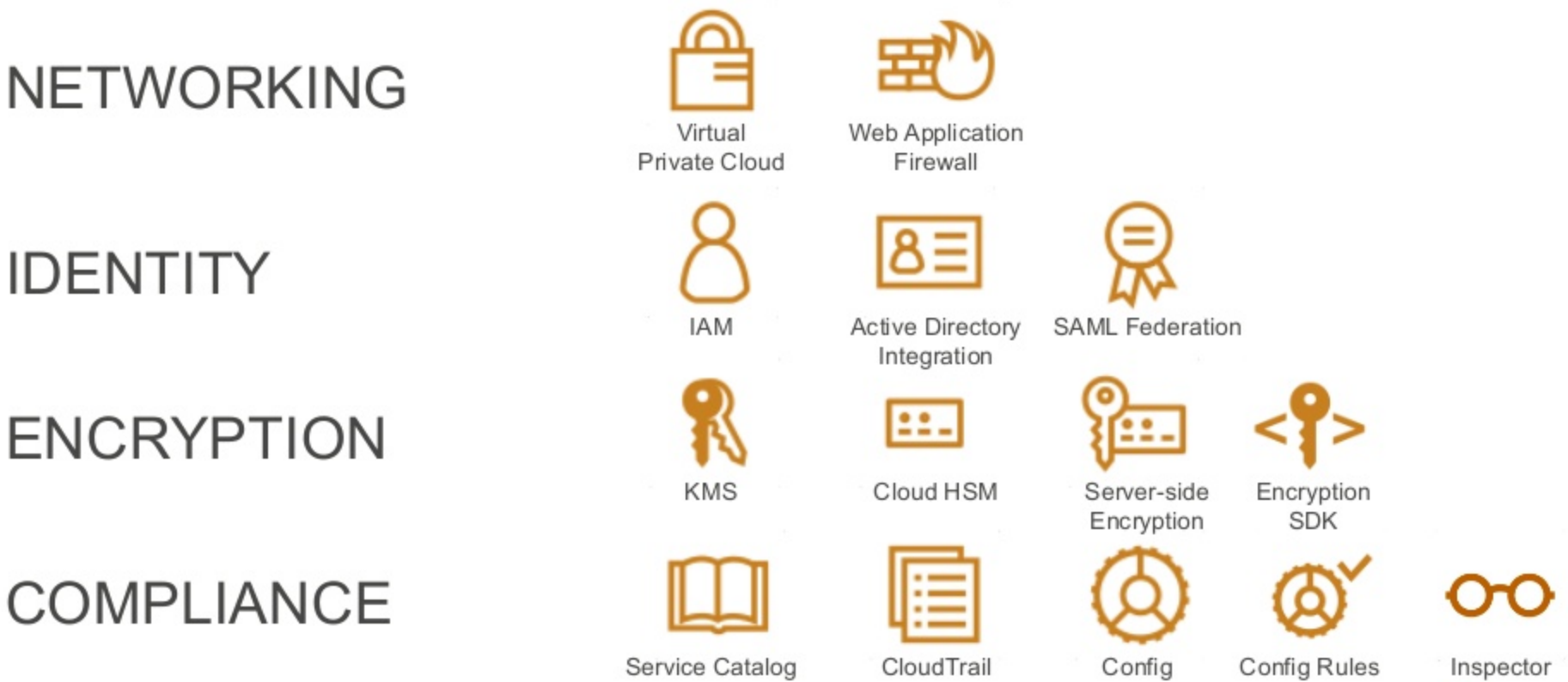
and prevent wrongdoing in the U.S. markets

Discipline

those who break the rules

FINRA handles approximately 75 billion market events every day to build a holistic picture of trading in the U.S.

Broadest Services to Secure Applications



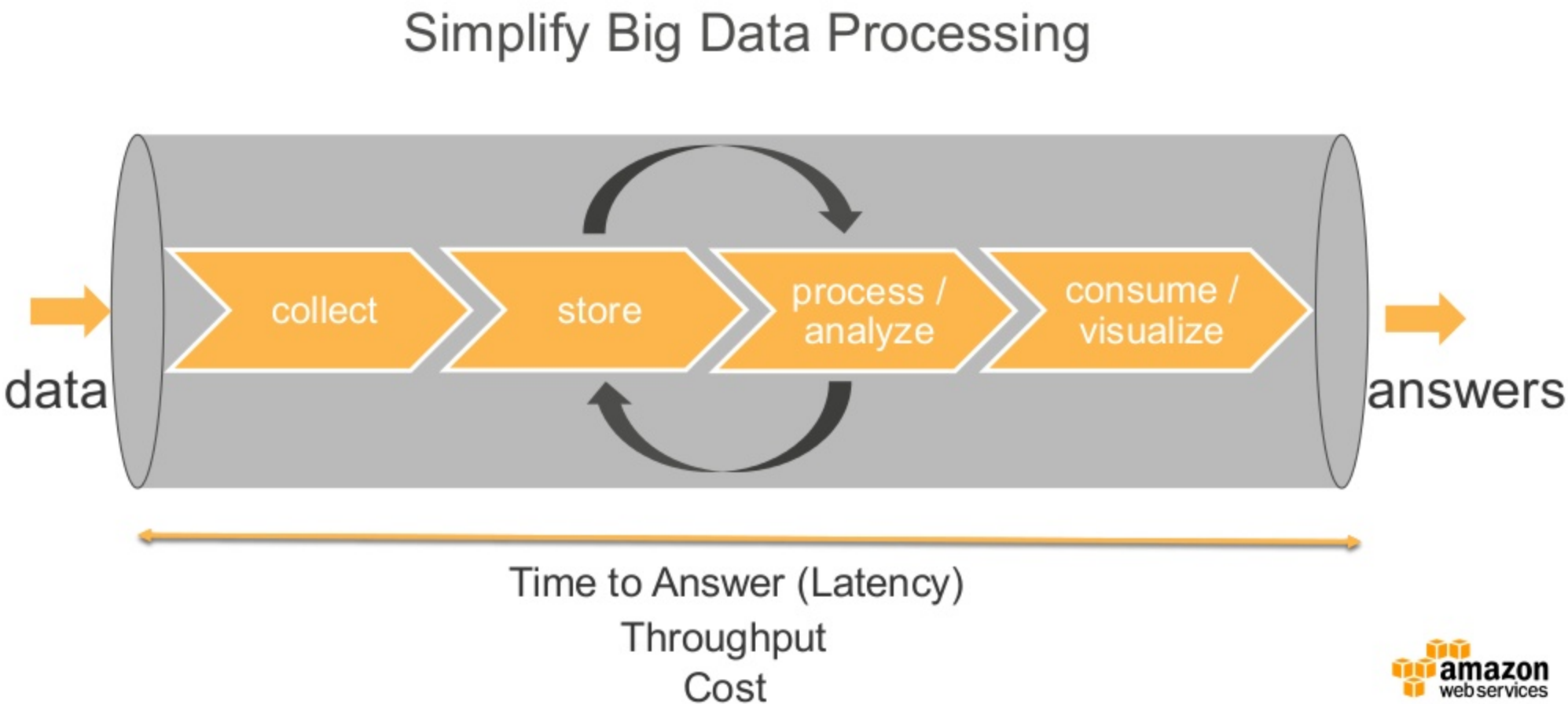
amazon web services

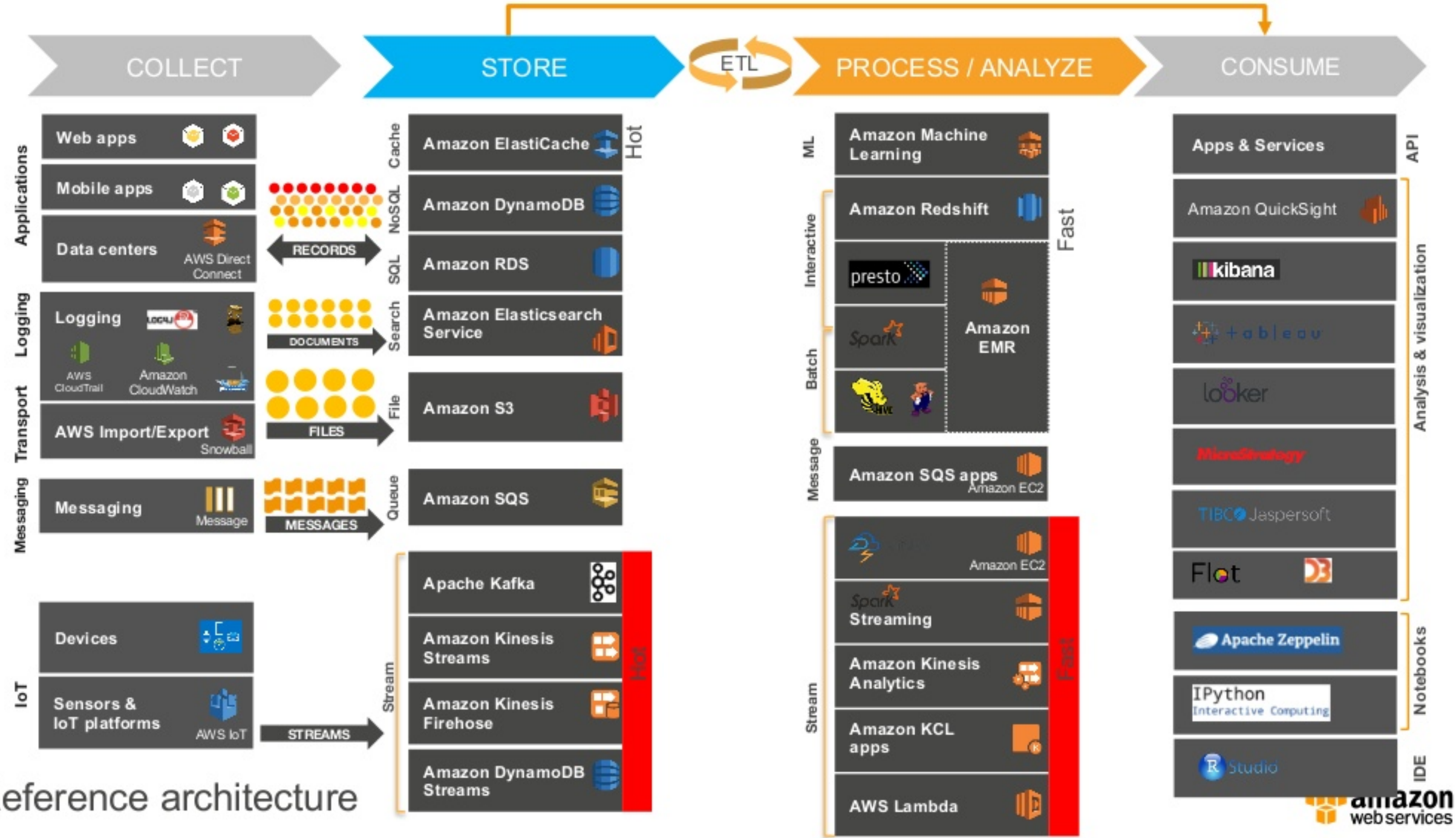
Key AWS Certifications and Assurance Programs

amazon web services

Design Patterns & Architectures

Amazon Web Services





Reference architecture

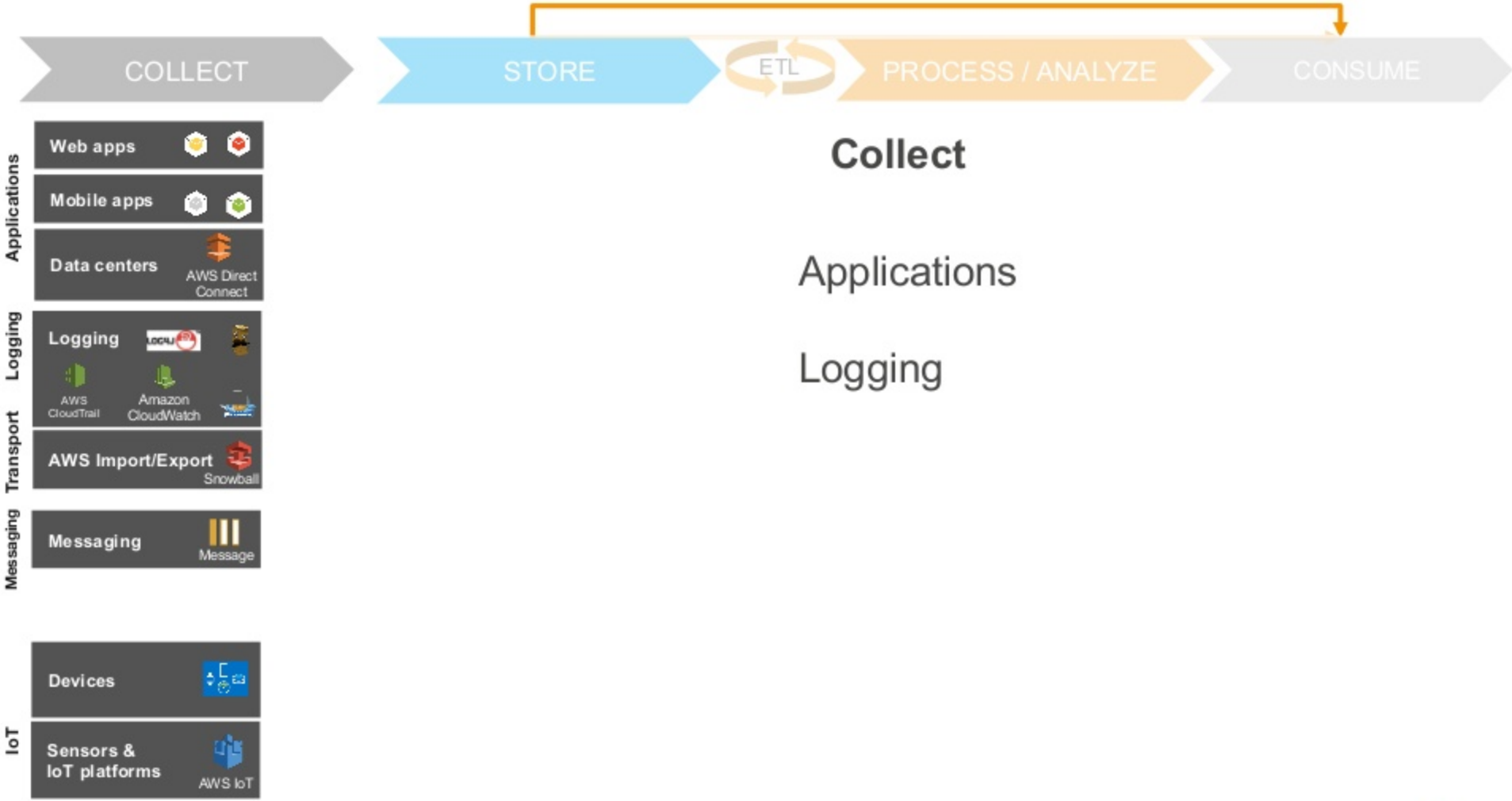
amazon web services

Reference architecture

amazon web services

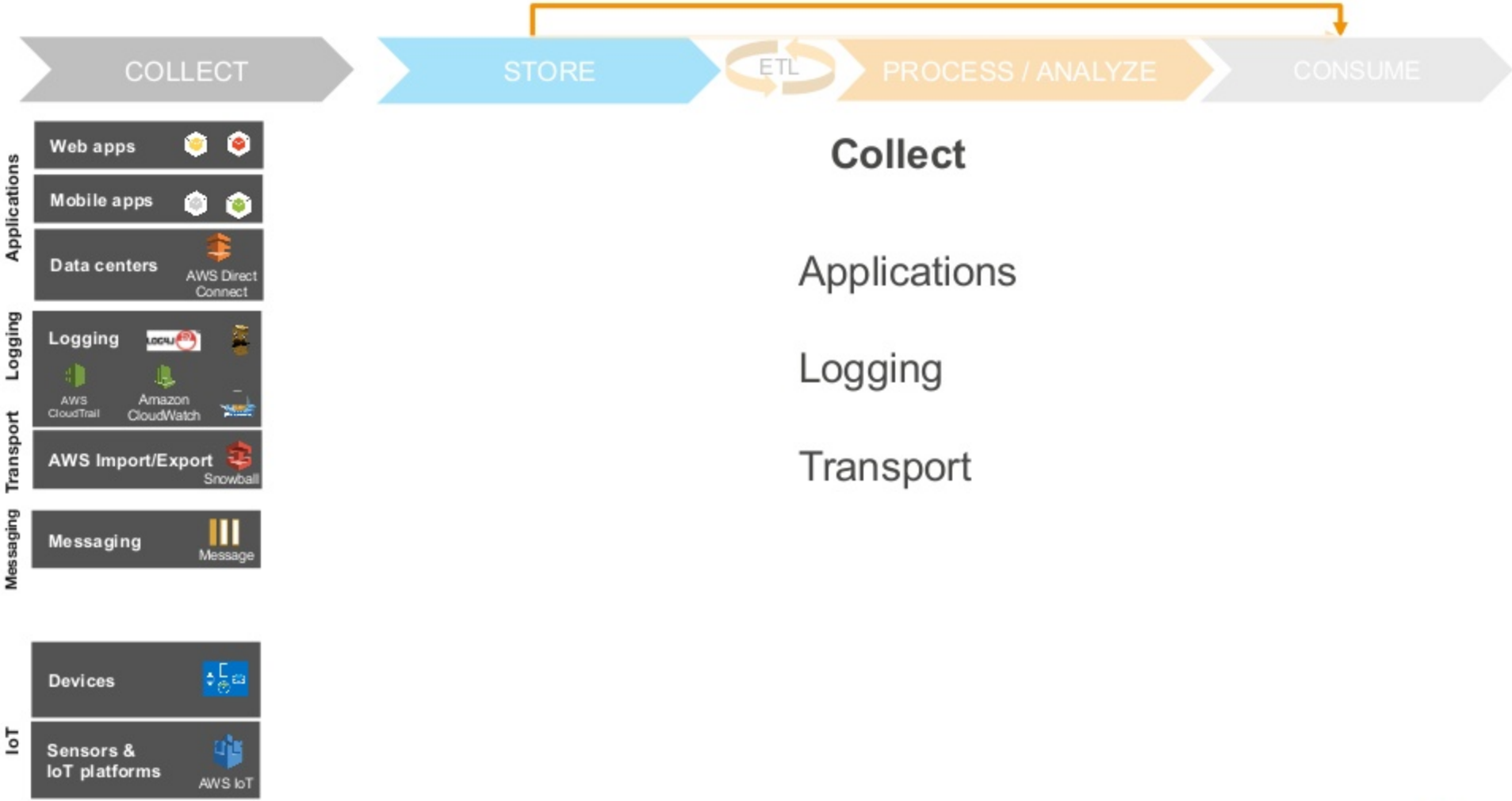
Reference architecture

amazon web services



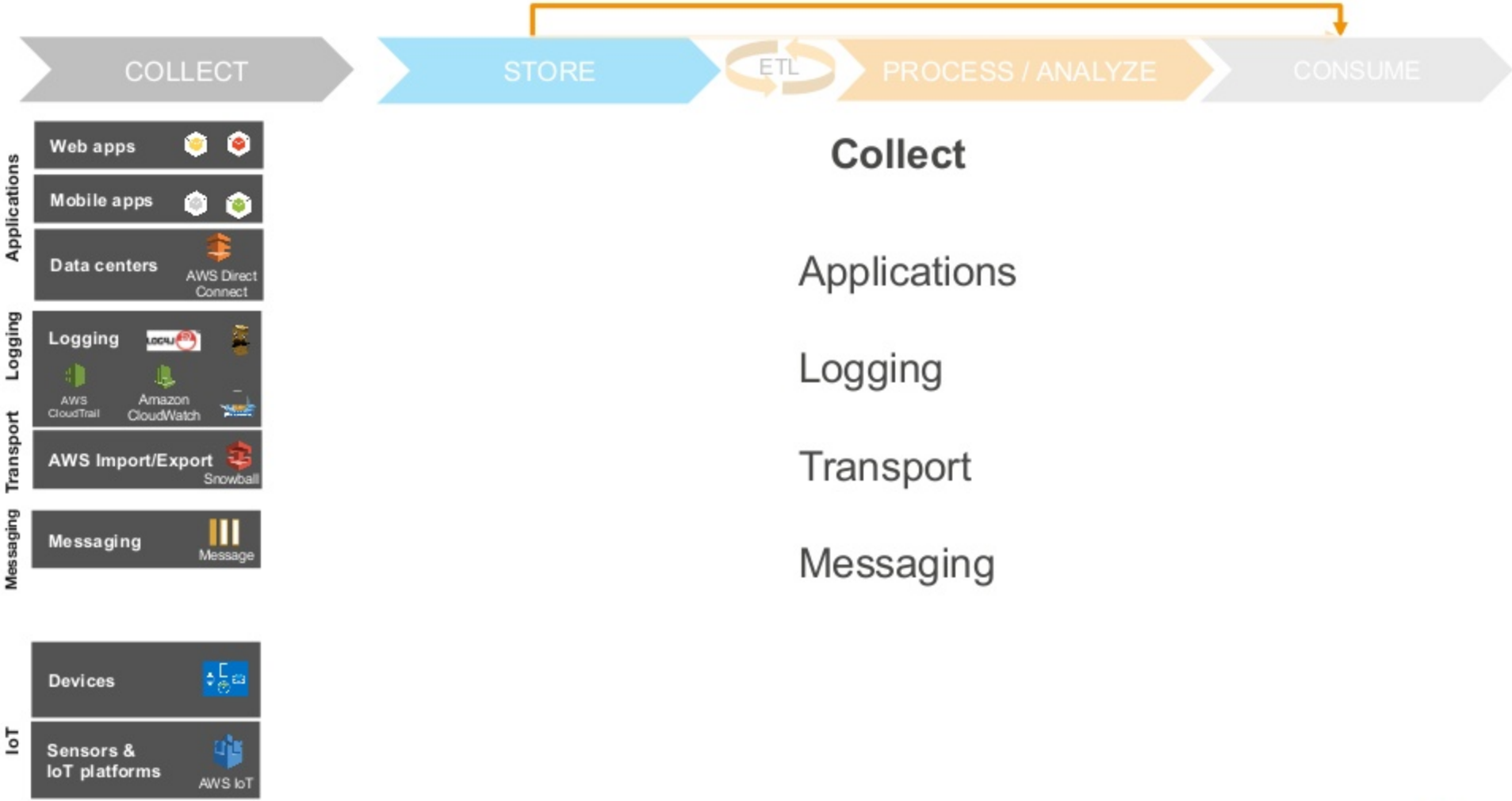
Reference architecture

amazon web services



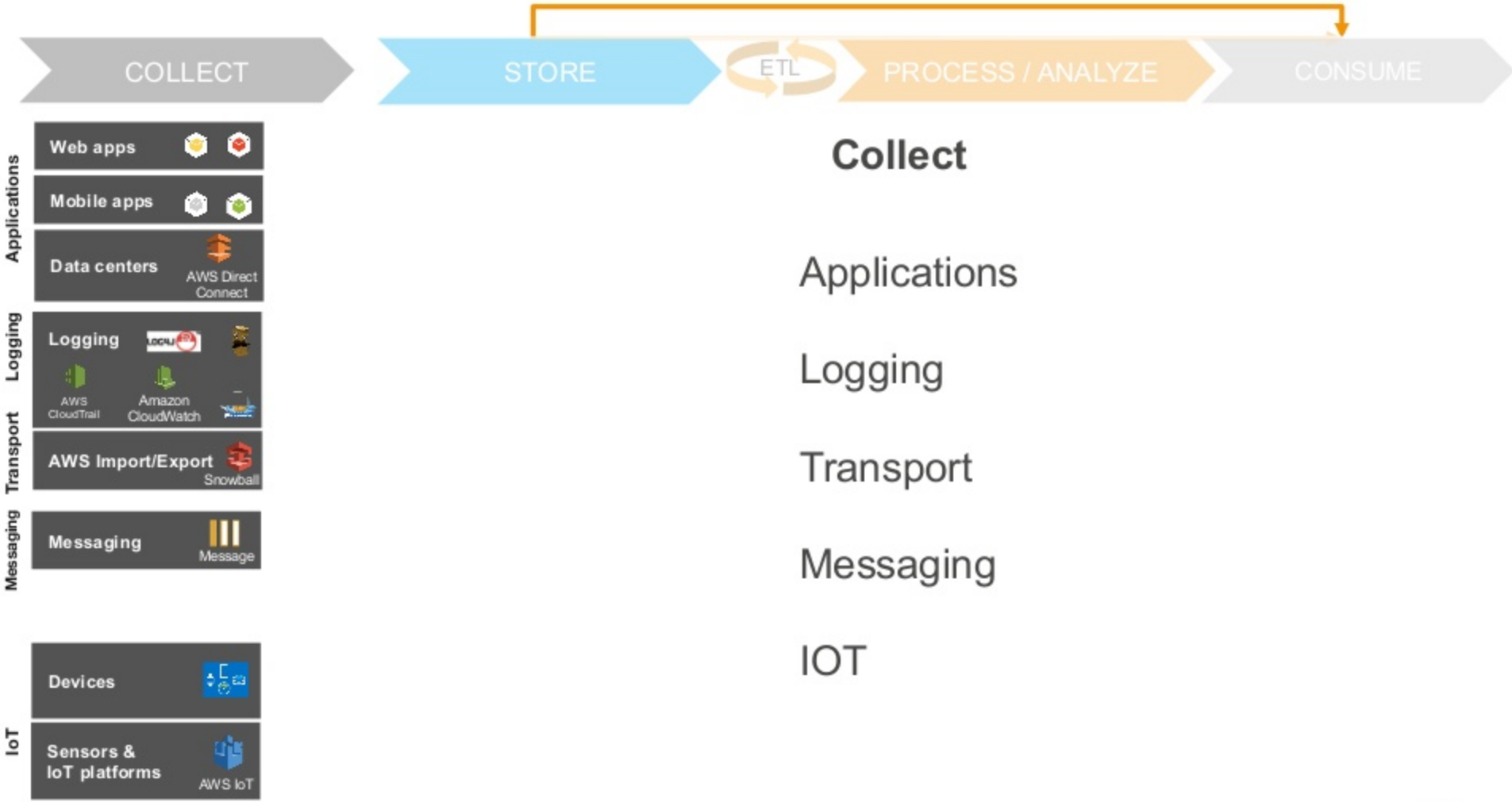
Reference architecture

amazon web services



Reference architecture

amazon web services



Reference architecture

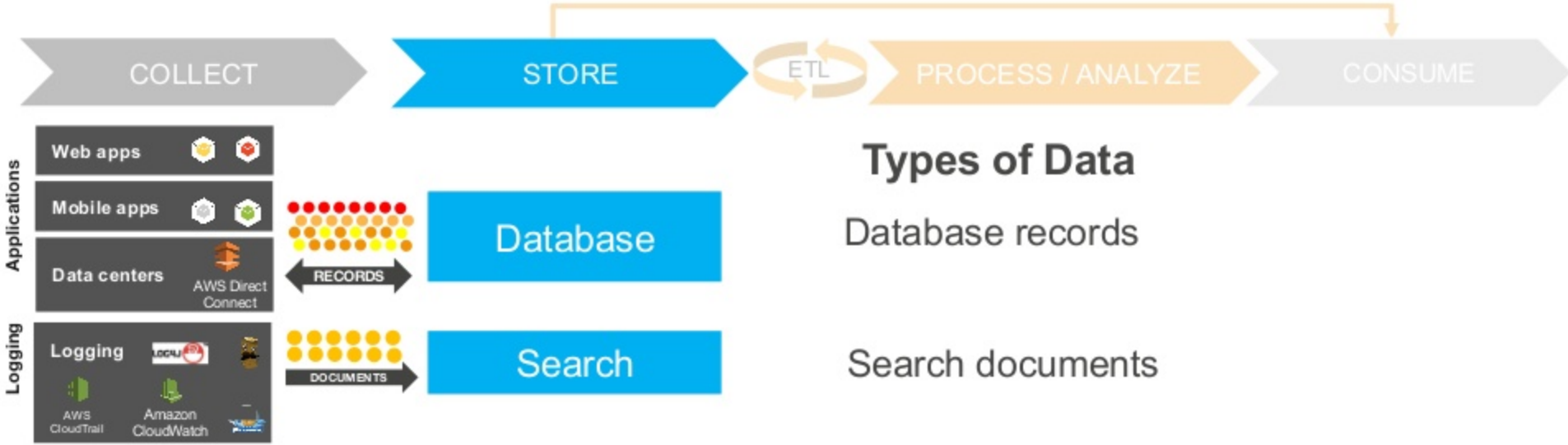
amazon web services

Reference architecture

amazon web services

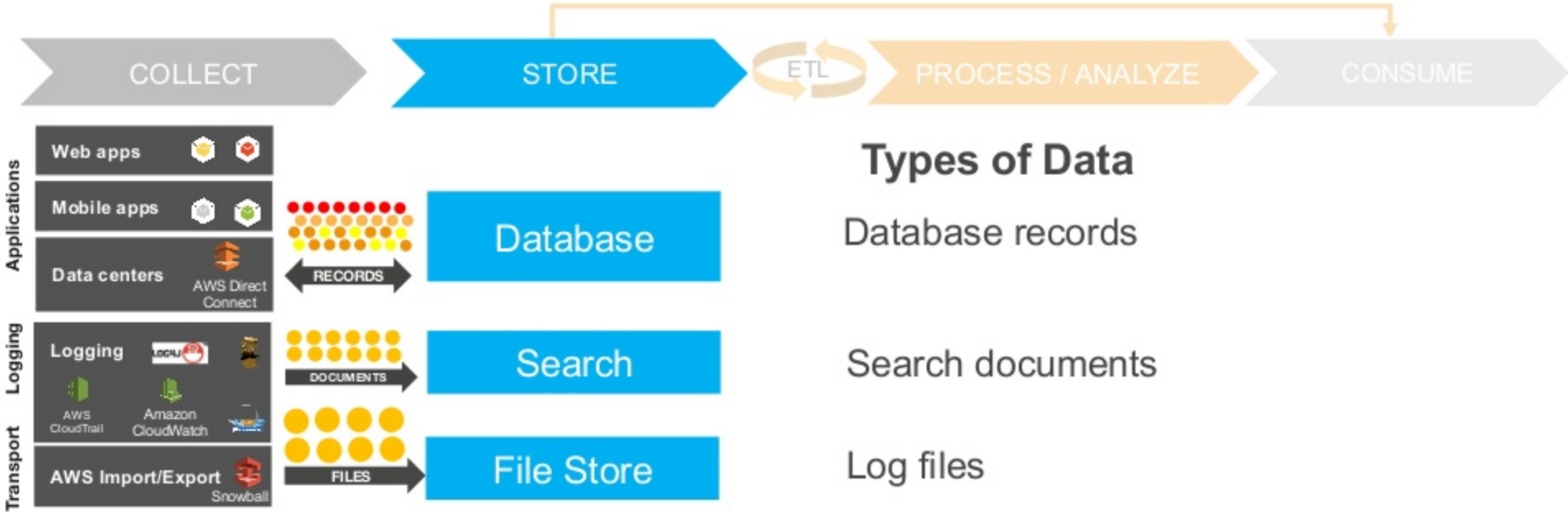
Reference architecture

amazon web services



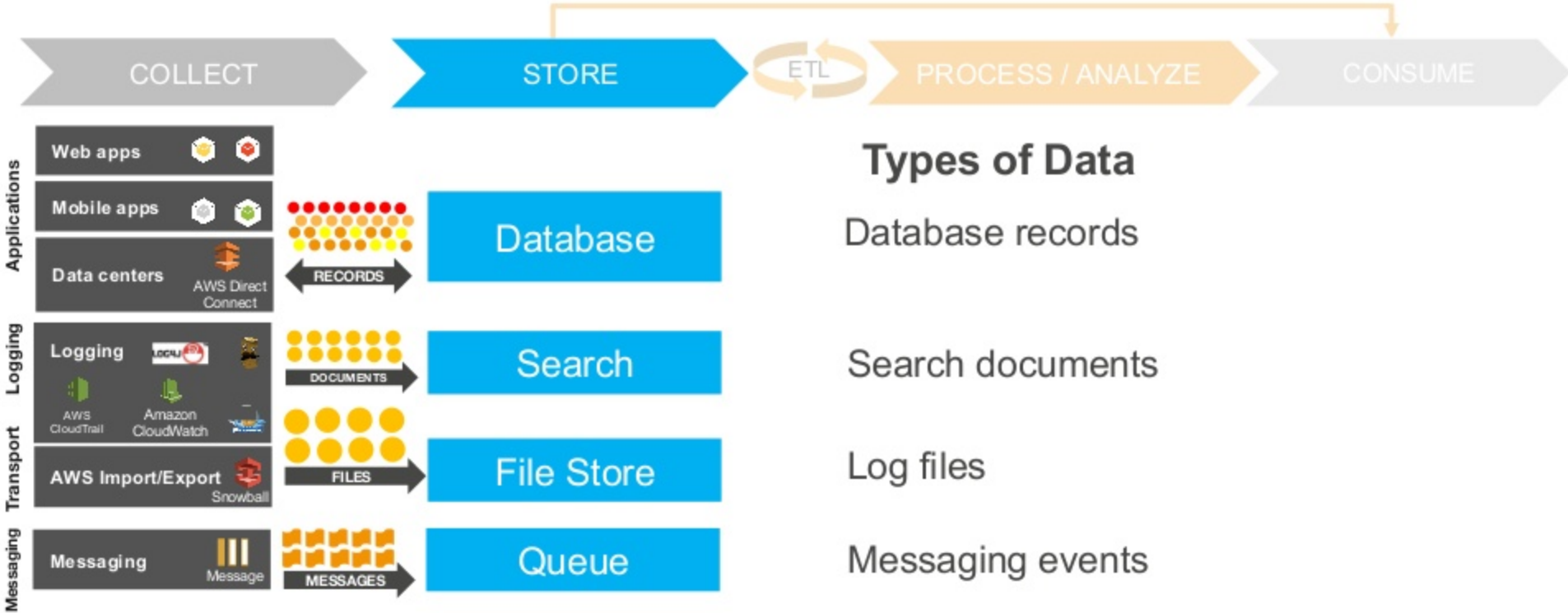
Reference architecture

amazon web services



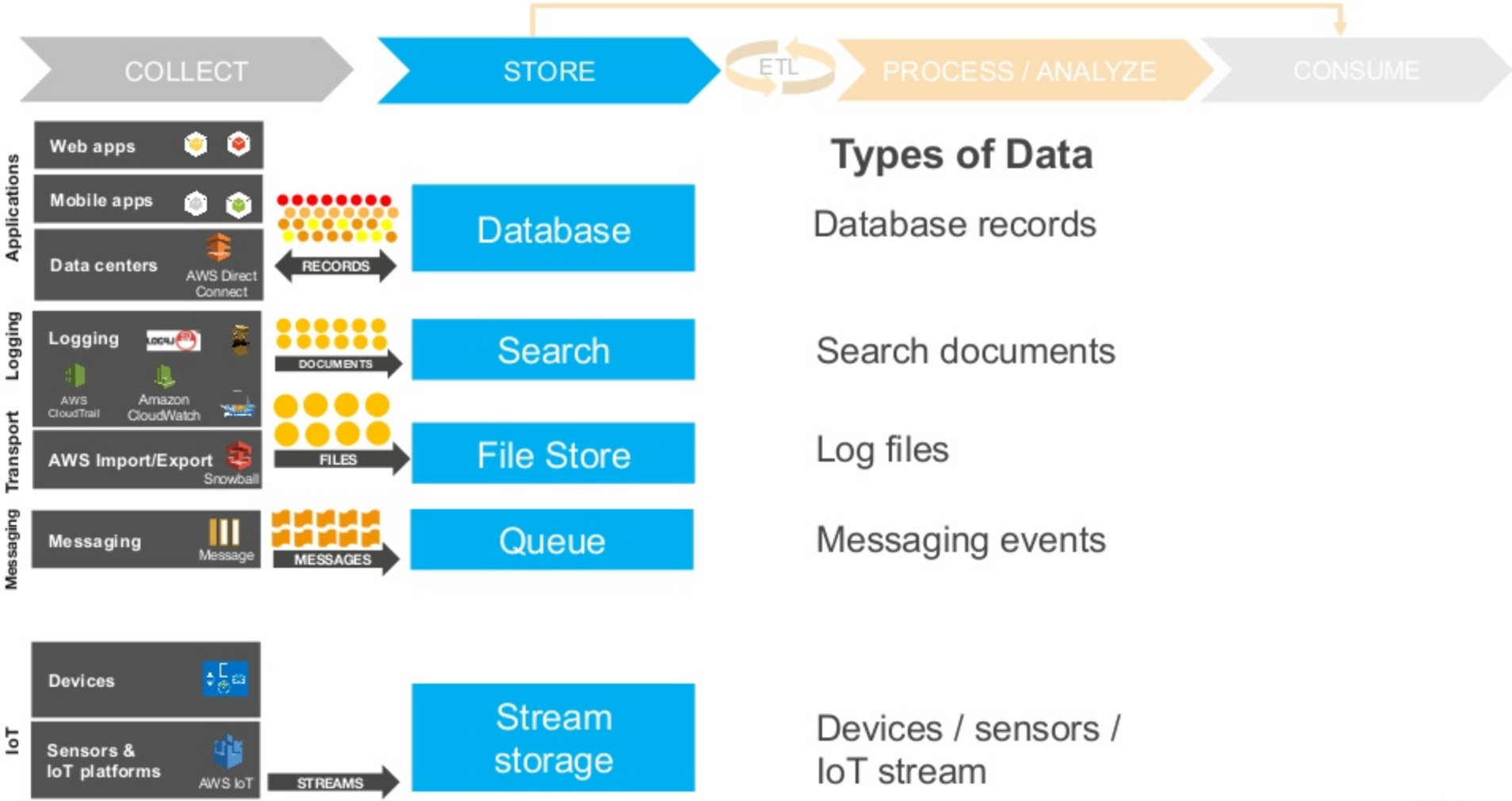
Reference architecture

amazon web services



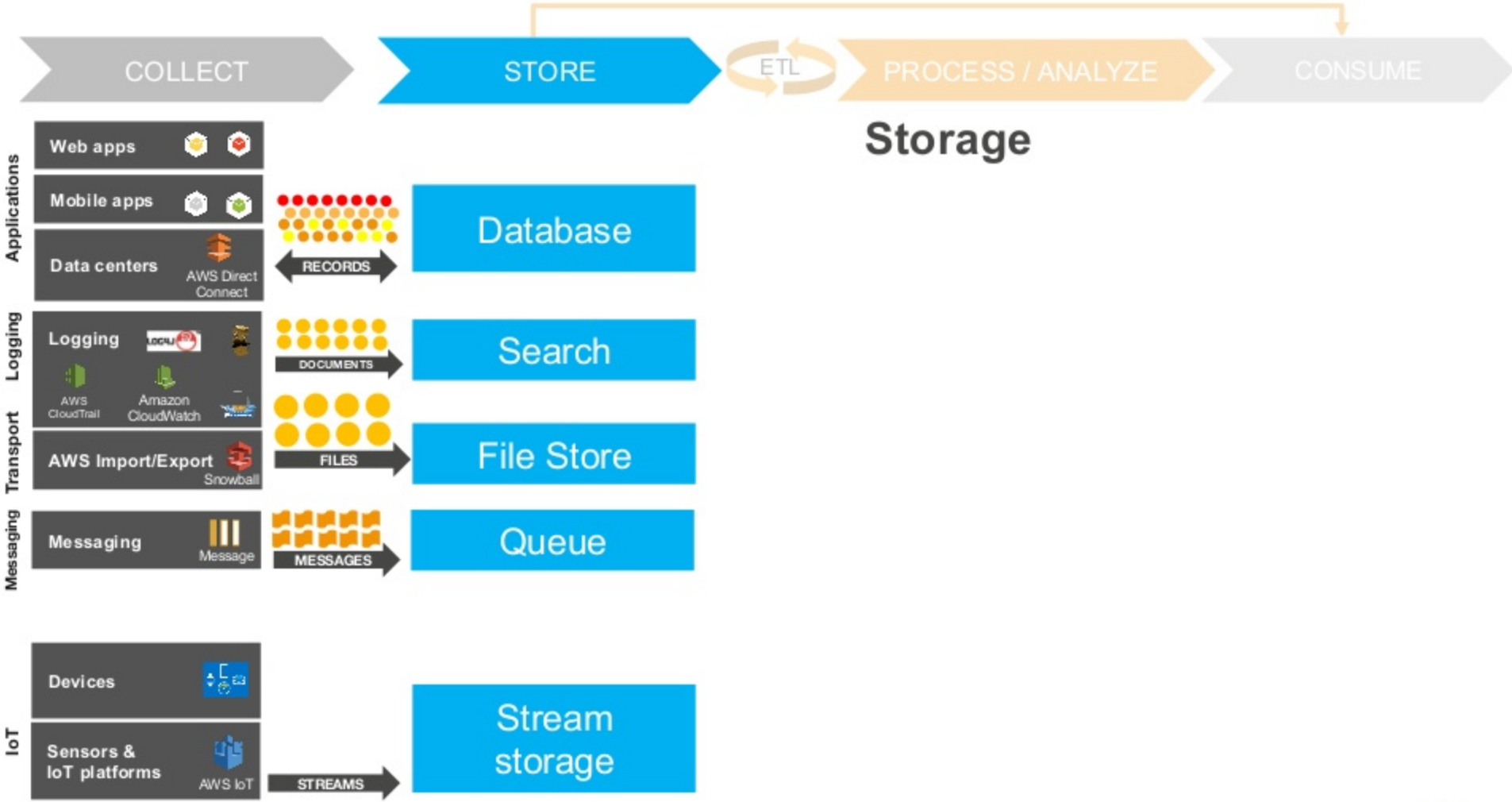
Reference architecture

amazon web services



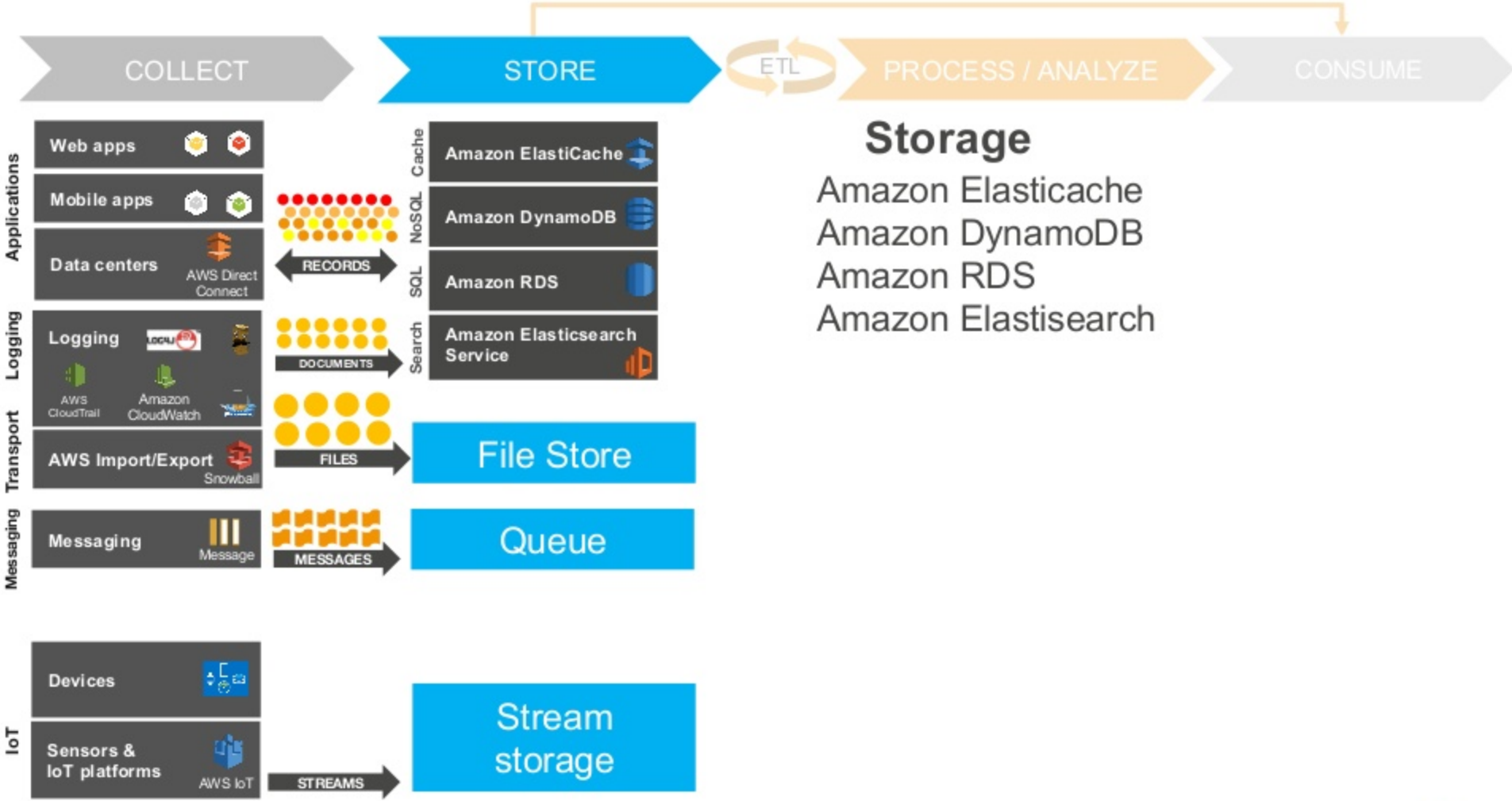
Reference architecture

amazon web services



Reference architecture

amazon web services

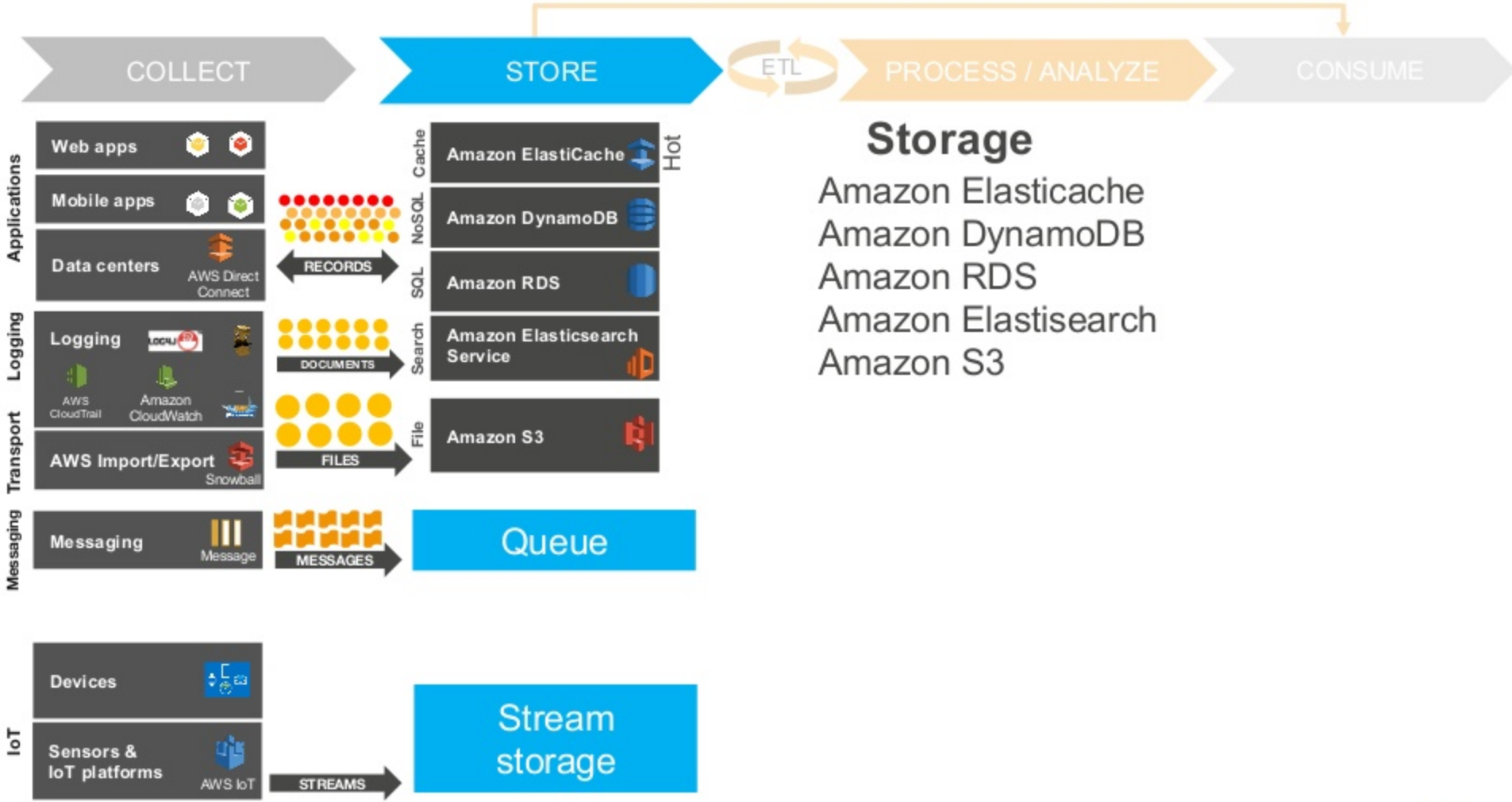


Storage

Amazon Elasticache
Amazon DynamoDB
Amazon RDS
Amazon Elastisearch

Reference architecture

amazon web services

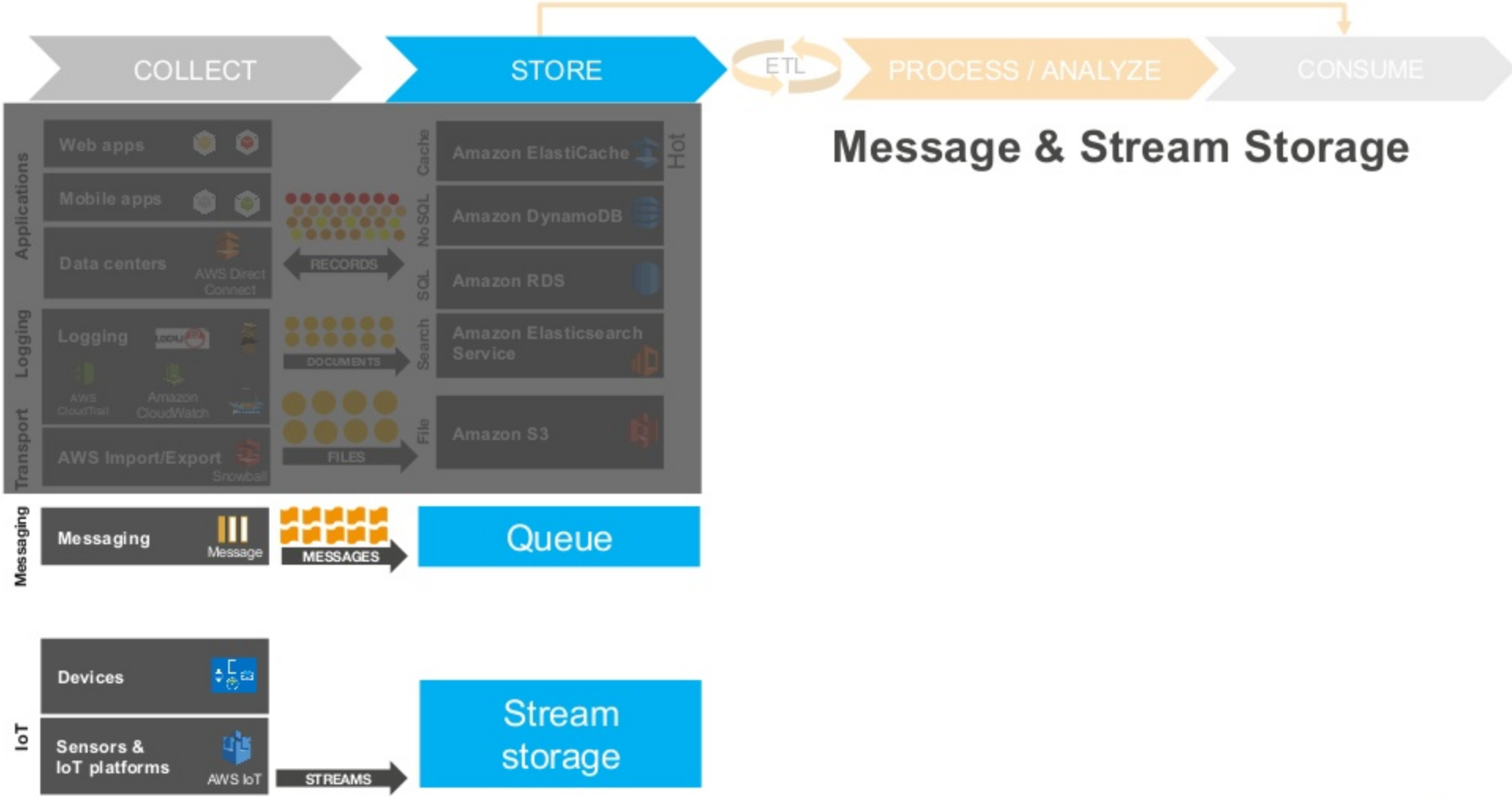


Storage

Amazon Elasticache
Amazon DynamoDB
Amazon RDS
Amazon Elastisearch
Amazon S3

Reference architecture

amazon web services



Reference architecture

amazon web services

Message & Stream Storage

Amazon SQS

- Managed message queue service

Reference architecture

amazon web services

Reference architecture

Message & Stream Storage

Amazon SQS

- Managed message queue service

Apache Kafka

- High throughput distributed messaging system

amazon web services

Reference architecture

Message & Stream Storage

Amazon SQS

- Managed message queue service

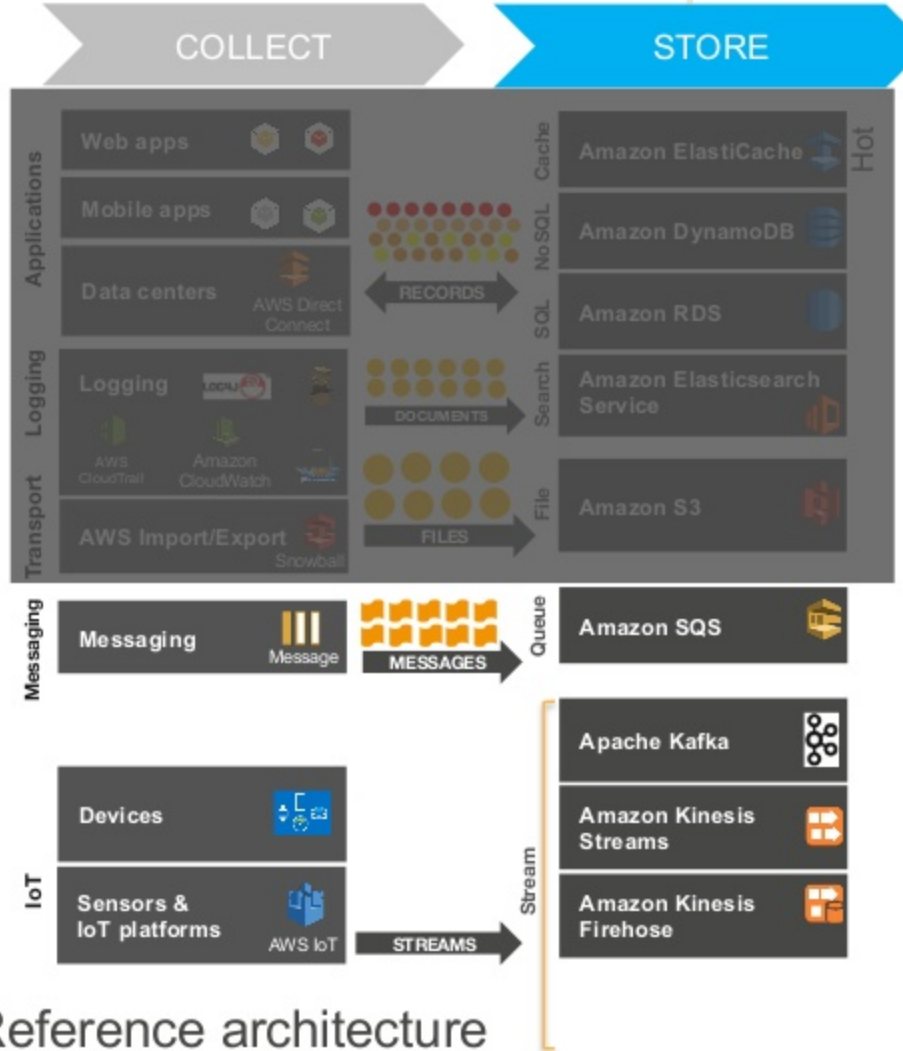
Apache Kafka

- High throughput distributed messaging system

Amazon Kinesis Streams

- Managed stream storage + processing

amazon web services



Reference architecture

Message & Stream Storage

Amazon SQS

- Managed message queue service

Apache Kafka

- High throughput distributed messaging system

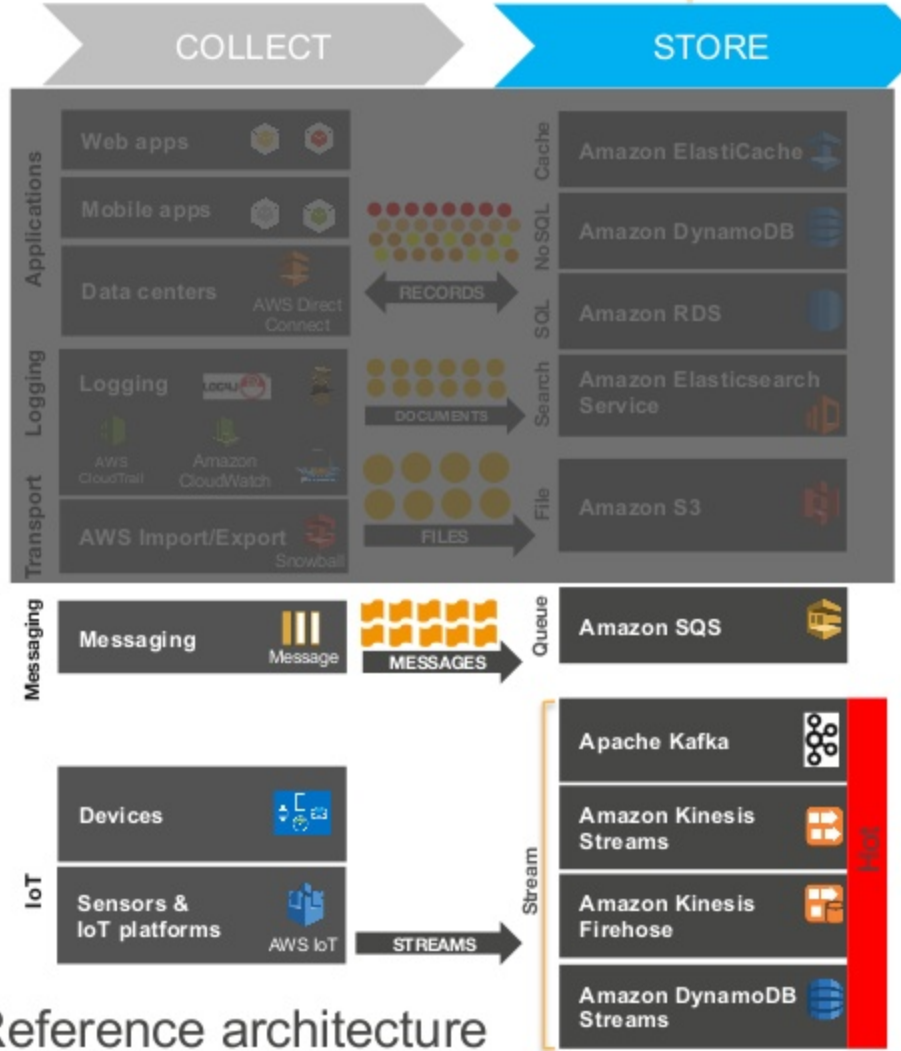
Amazon Kinesis Streams

- Managed stream storage + processing

Amazon Kinesis Firehose

- Managed data delivery

amazon web services



Reference architecture

Message & Stream Storage

Amazon SQS
- Managed message queue service

Apache Kafka
- High throughput distributed messaging system

Amazon Kinesis Streams
- Managed stream storage + processing

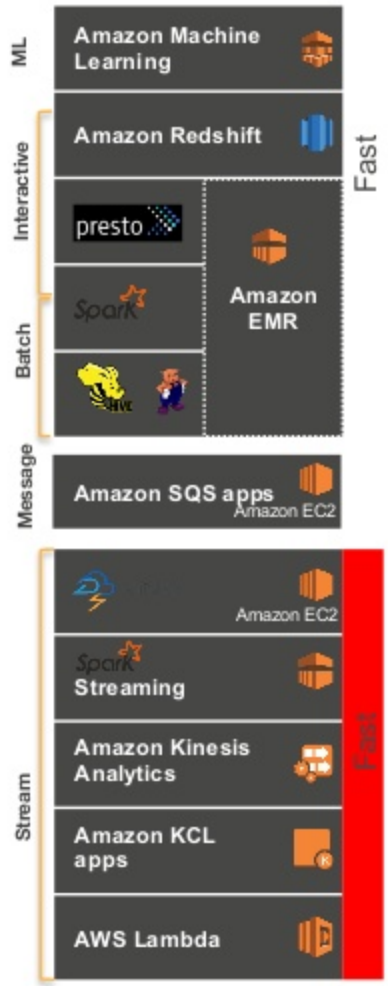
Amazon Kinesis Firehose
- Managed data delivery

Amazon DynamoDB
- Managed NoSQL database
- Tables can be stream-enabled

amazon web services

COLLECT | STORE | ETL | PROCESS / ANALYZE

Tools and Frameworks



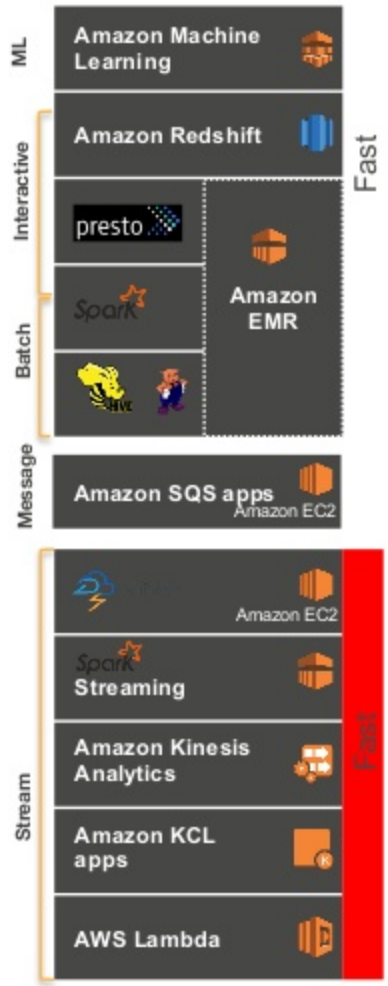
Reference architecture

amazon web services

COLLECT | STORE | ETL | PROCESS / ANALYZE

Tools and Frameworks

Machine Learning
- Amazon ML, Amazon EMR (Spark ML)



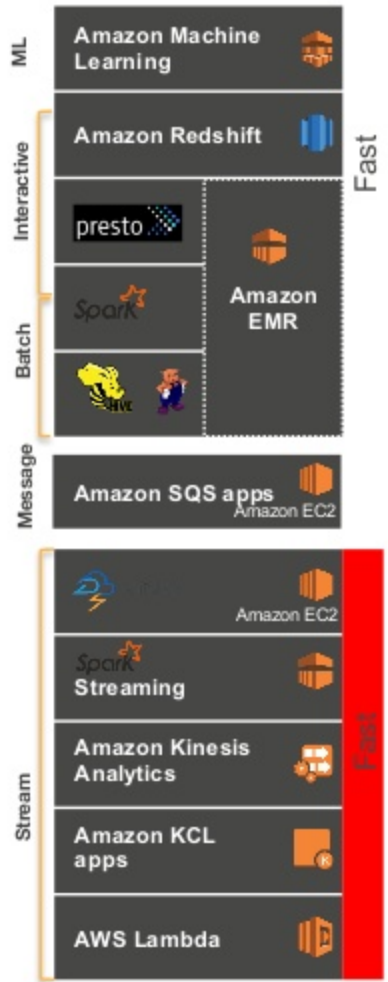
Reference architecture

amazon web services

Tools and Frameworks

Machine Learning
- Amazon ML, Amazon EMR (Spark ML)

Interactive
- Amazon Redshift, Amazon EMR (Presto, Spark)



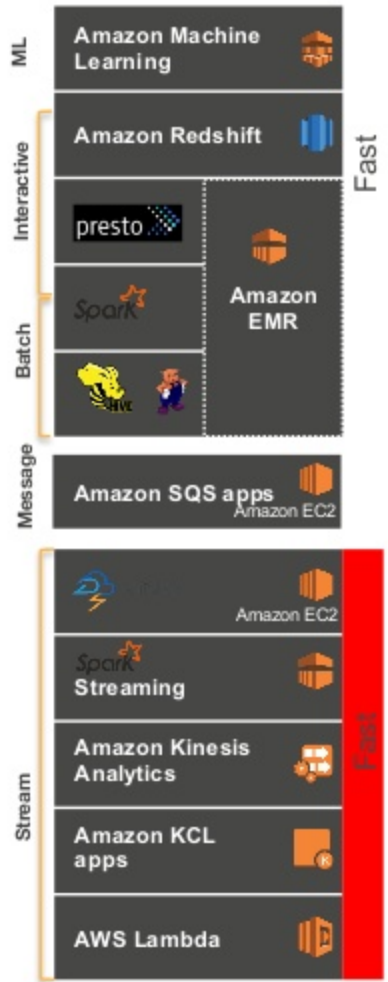
Reference architecture

Tools and Frameworks

Machine Learning
- Amazon ML, Amazon EMR (Spark ML)

Interactive
- Amazon Redshift, Amazon EMR (Presto, Spark)

Batch
- Amazon EMR (MapReduce, Hive, Pig, Spark)



Reference architecture

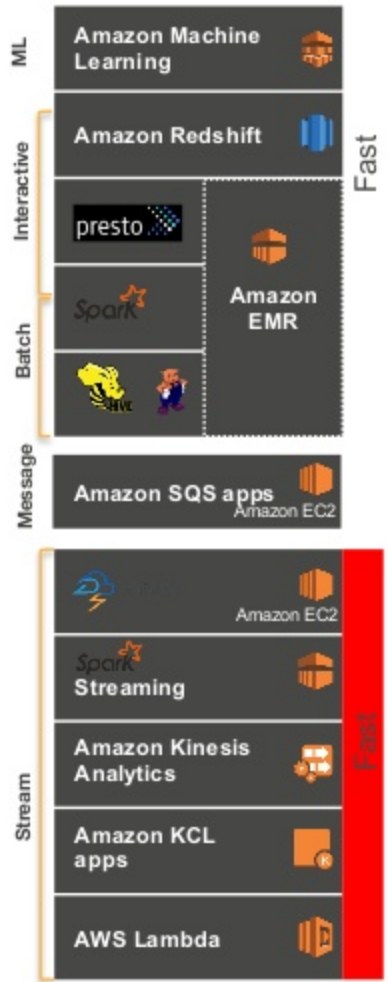
Tools and Frameworks

Machine Learning
- Amazon ML, Amazon EMR (Spark ML)

Interactive
- Amazon Redshift, Amazon EMR (Presto, Spark)

Batch
- Amazon EMR (MapReduce, Hive, Pig, Spark)

Messaging
- Amazon SQS application on Amazon EC2



Reference architecture

amazon web services

Tools and Frameworks

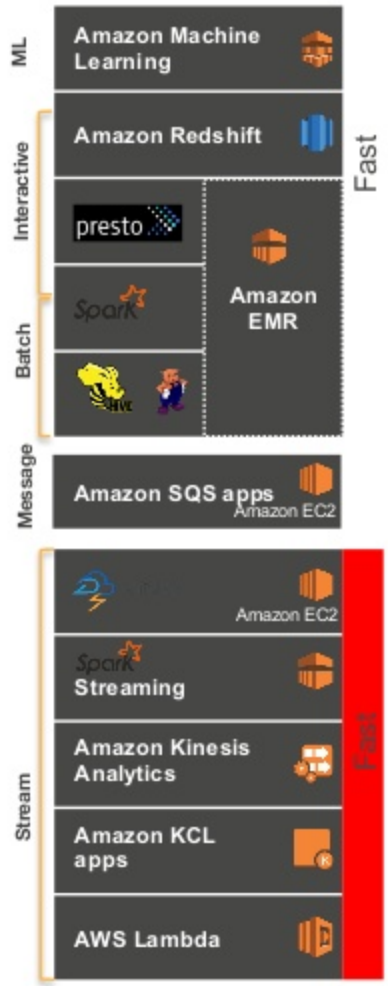
Machine Learning
- Amazon ML, Amazon EMR (Spark ML)

Interactive
- Amazon Redshift, Amazon EMR (Presto, Spark)

Batch
- Amazon EMR (MapReduce, Hive, Pig, Spark)

Messaging
- Amazon SQS application on Amazon EC2

Streaming
- Micro-batch: Spark Streaming, KCL
- Real-time: Amazon Kinesis Analytics, Storm, AWS Lambda, KCLç



Reference architecture

Consume

Reference architecture

Consume

Applications & API

Reference architecture

amazon web services

Consume

Applications & API

Analysis and visualization

Reference architecture

amazon web services

Consume

Applications & API

Analysis and visualization

Notebooks

Reference architecture

amazon web services

Consume

Applications & API

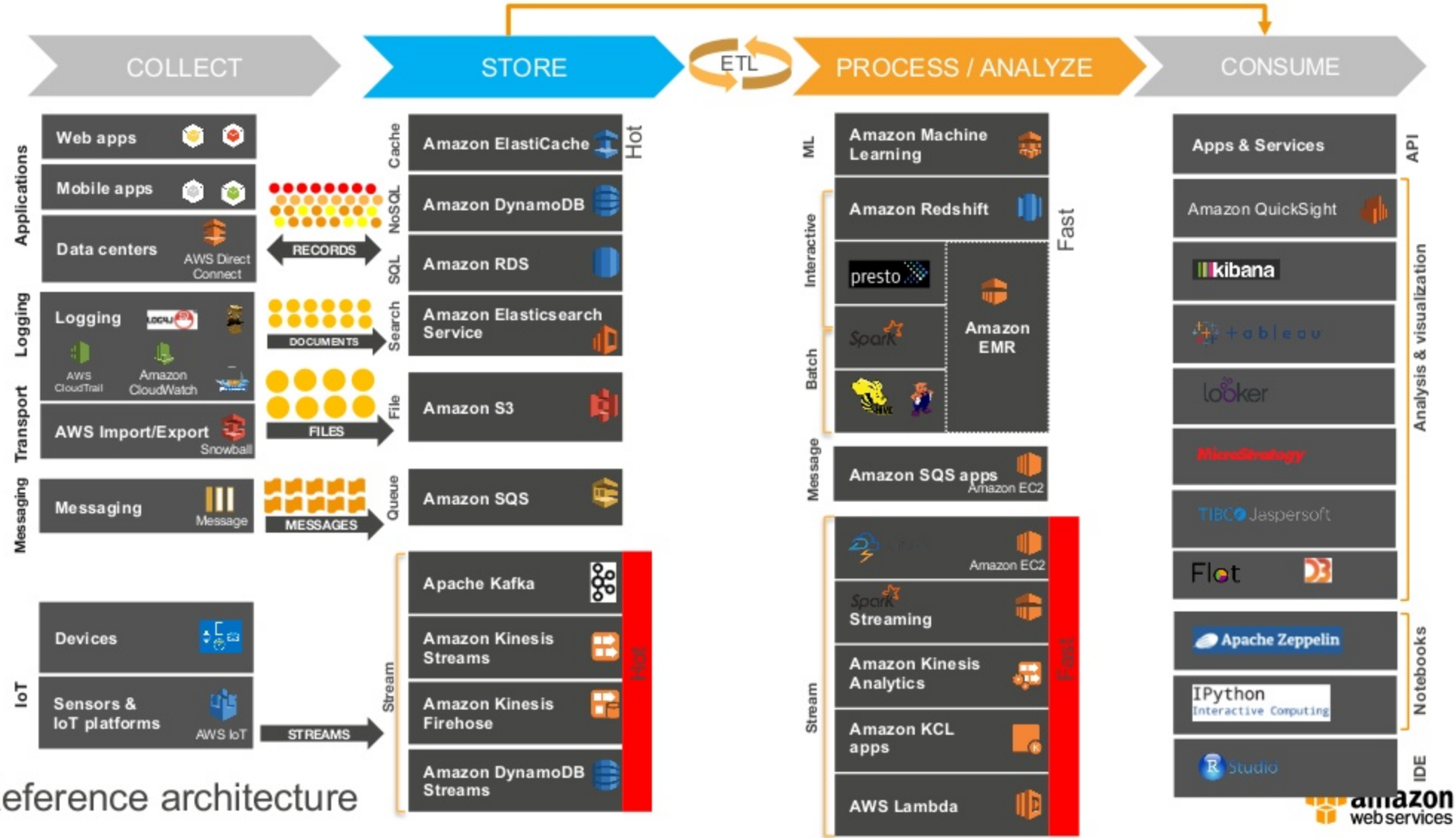
Analysis and visualization

Notebooks

IDE

Reference architecture

amazon web services

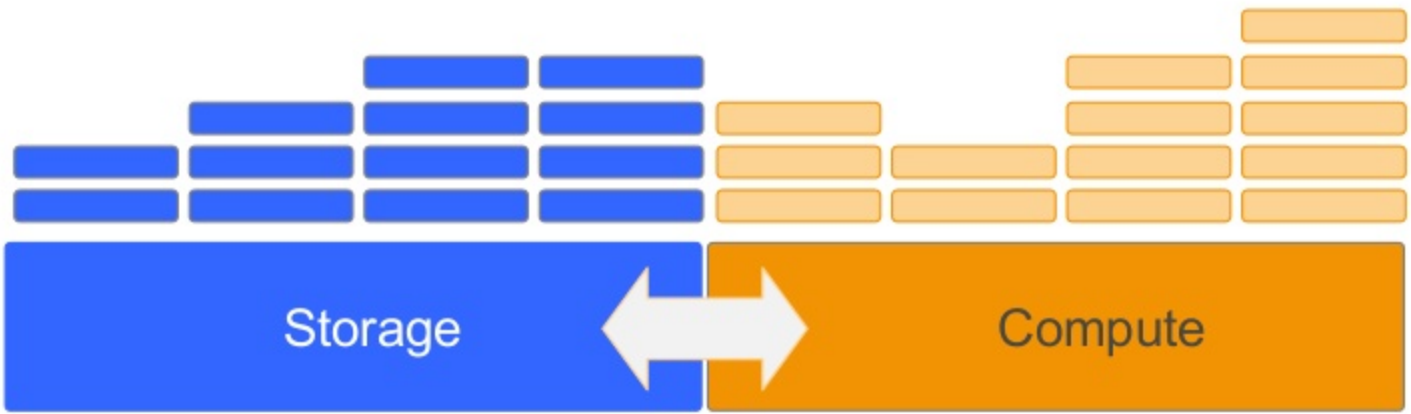


Reference architecture

amazon web services

Multi-Stage Decoupled “Data Bus”

- Decoupled “data bus”
  - Completely autonomous components
  - Independently scale storage & compute resources on-demand



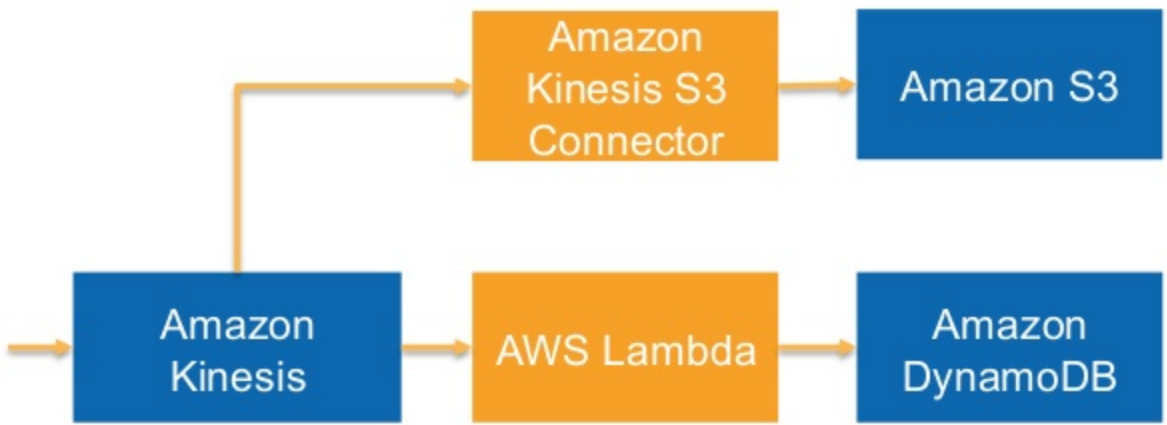
Collect → Store → Process | Analyze → Consume

Multi-Stage Decoupled “Data Bus”

Multiple stages

Storage decouples multiple processing stages

Multiple Stream Processing Applications Can Read from Amazon Kinesis

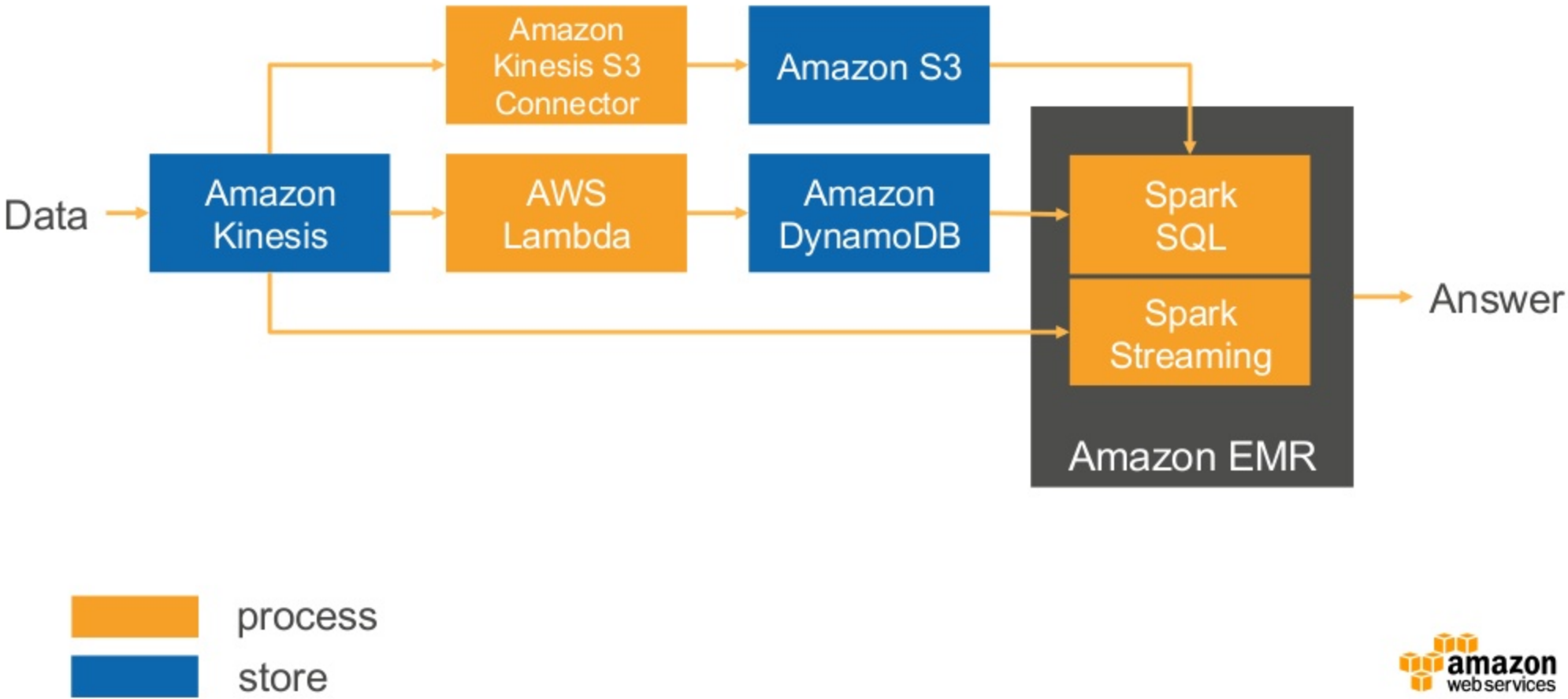


process

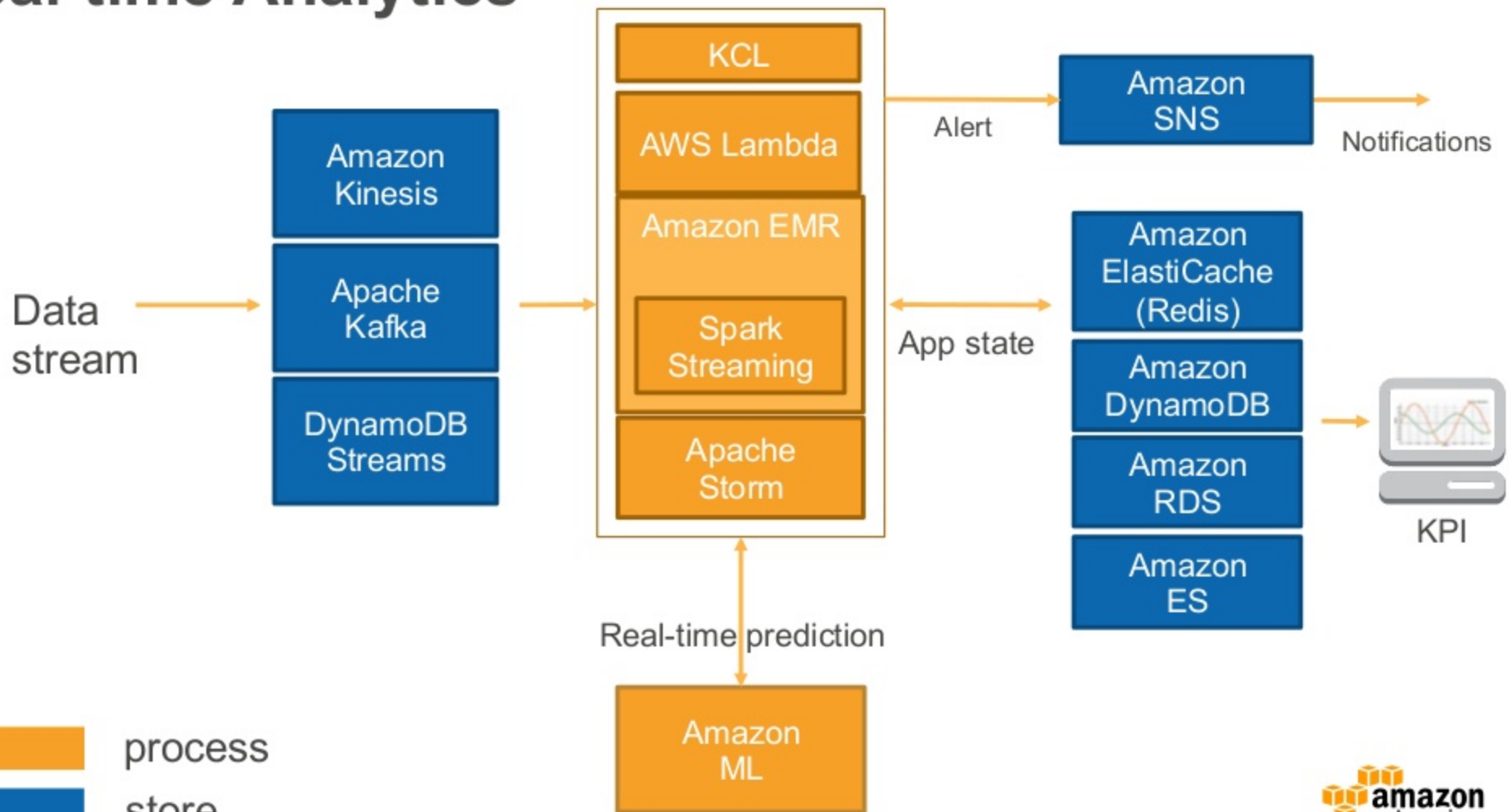
store

amazon web services

Analysis Frameworks Can Read from Multiple Data Stores



Real-time Analytics



Case Study: Clickstream Analysis

Send clickstream data to Kinesis Streams

Kinesis Streams stores and exposes clickstream data for processing

Custom application built on Kinesis Client Library makes real-time content recommendations

Readers see personalized content suggestions

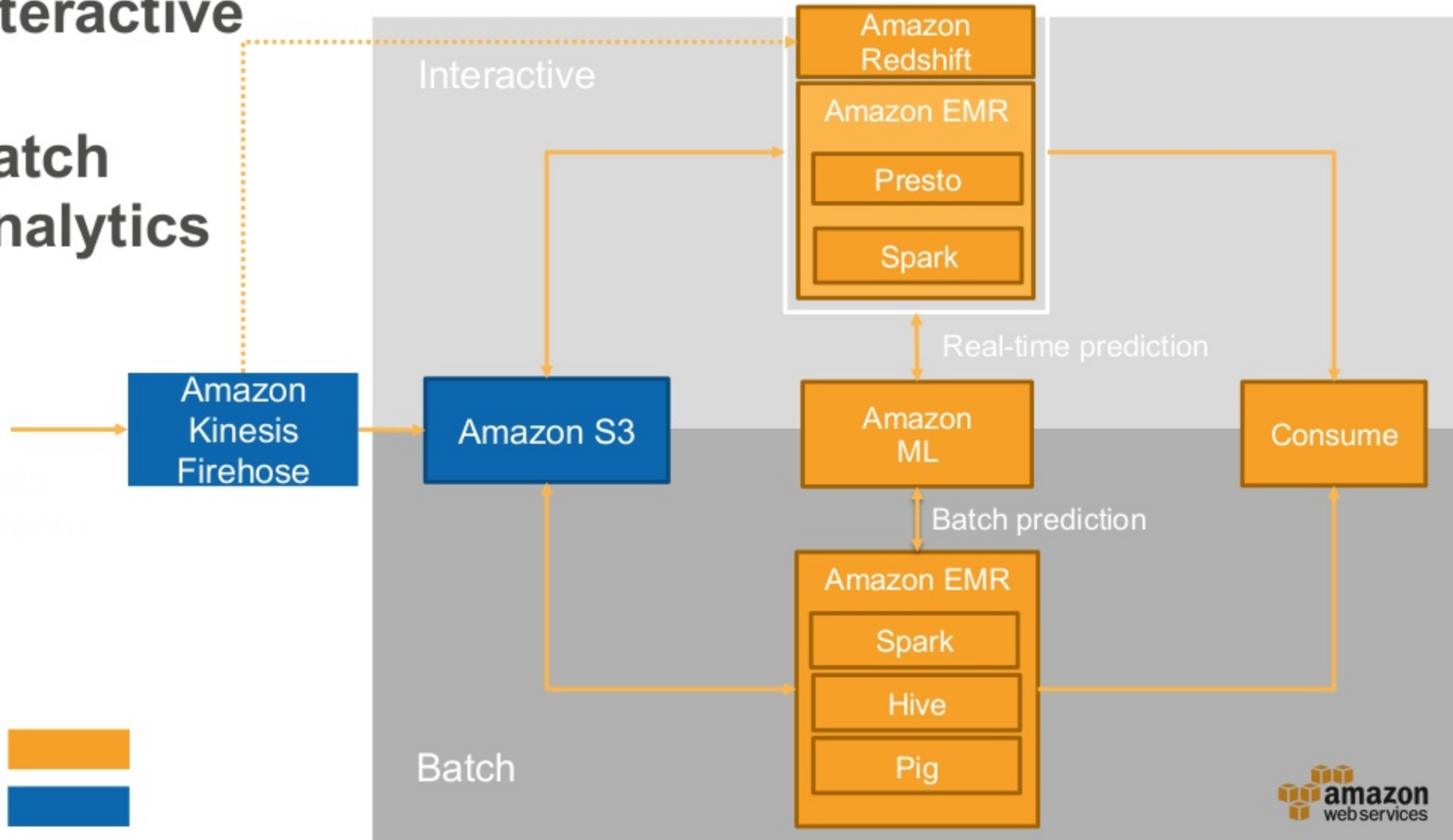
Store → Process | Analyze → Answers

HEARST

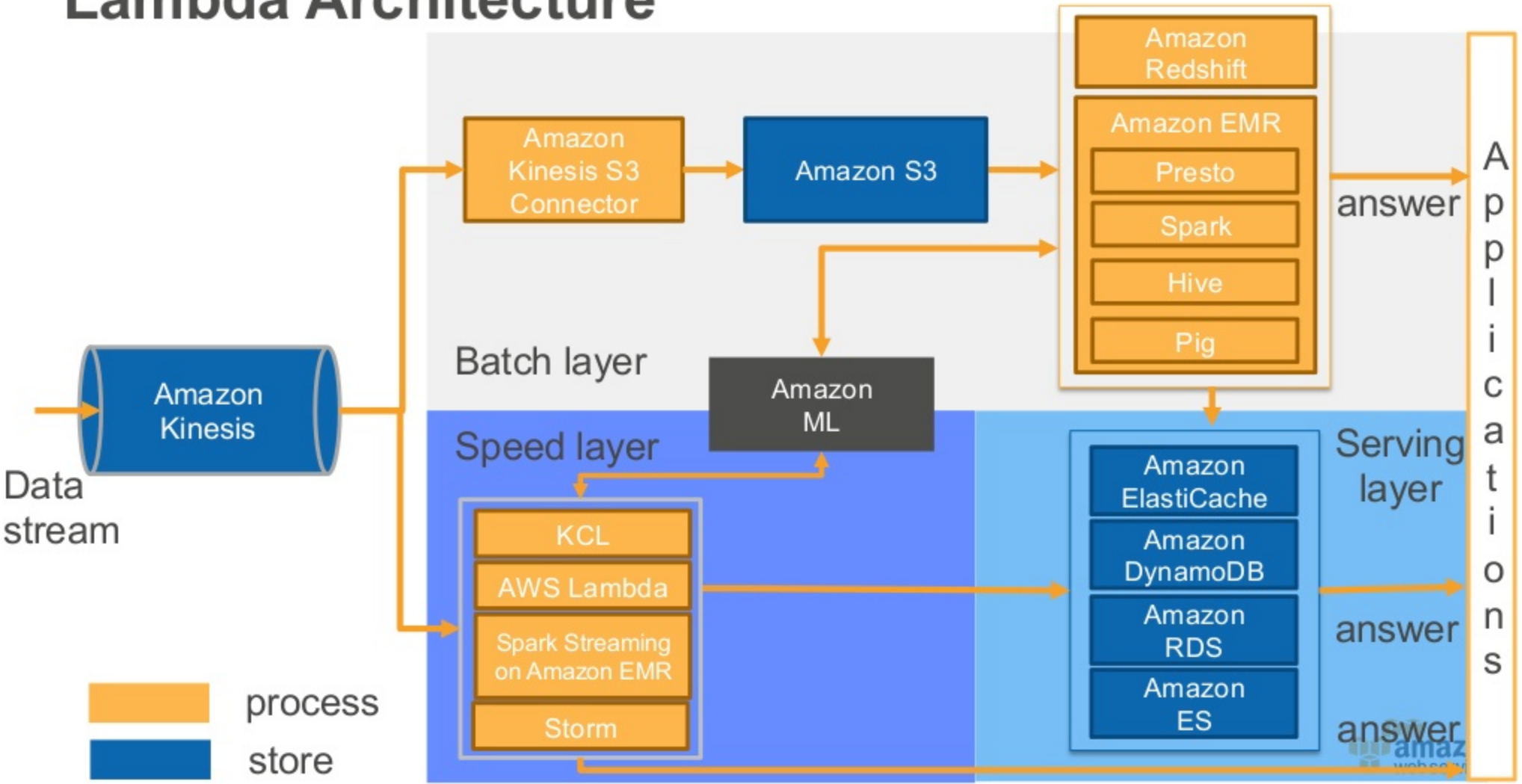
Hearst Corporation monitors trending content for over 250 digital properties worldwide and processes more than 30TB of data per day, using an architecture that includes Amazon Kinesis and Spark running on Amazon EMR.

amazon web services

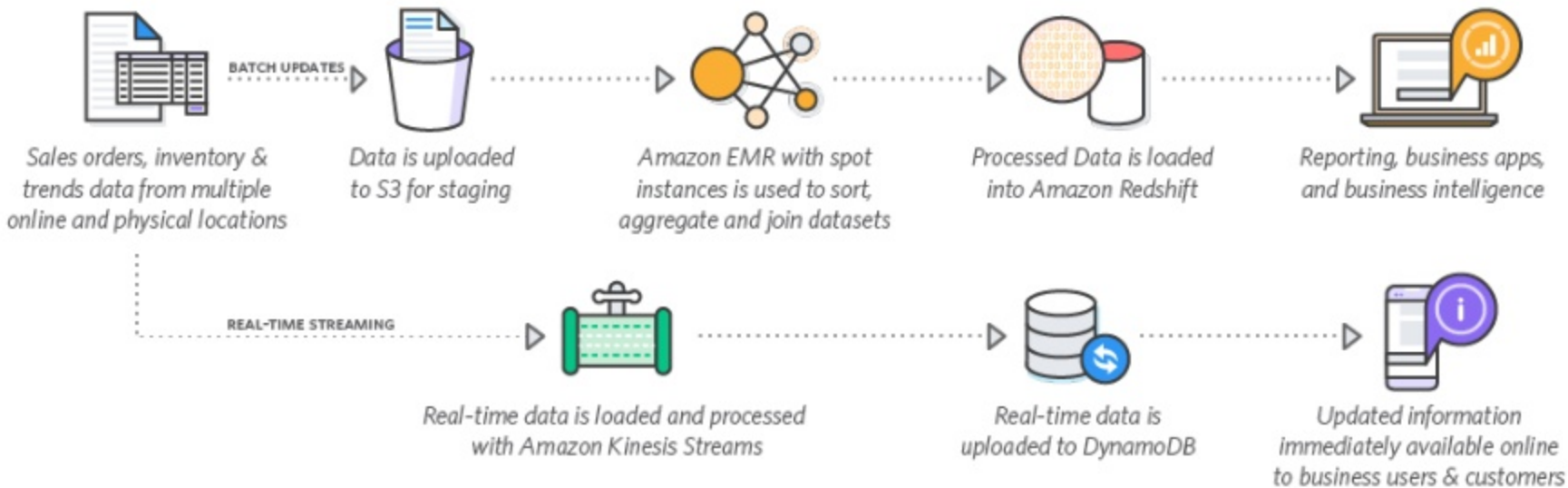
Interactive & Batch Analytics



Lambda Architecture



Case Study: On-demand Big Data Analytics



Store → Process → Store → Analyze → Answers

REDFIN

Redfin uses Amazon EMR with spot instances – dynamically spinning up & down Apache Hadoop clusters – to perform large data transformations and deliver data to internal and external customers.

amazon web services

Architectural Principles

Decoupled “data bus”

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

Use the right tool for the job

- Data structure, latency, throughput, access patterns

Use Lambda architecture ideas

- Immutable (append-only) log, batch/speed/serving layer

Leverage AWS managed services

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

Big data ≠ big cost

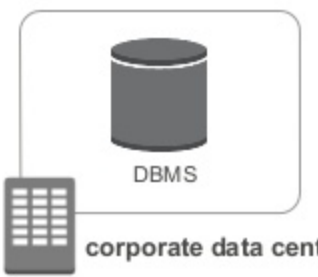
Building a Big Data Application

Building a Big Data Application

Getting Started

web clients

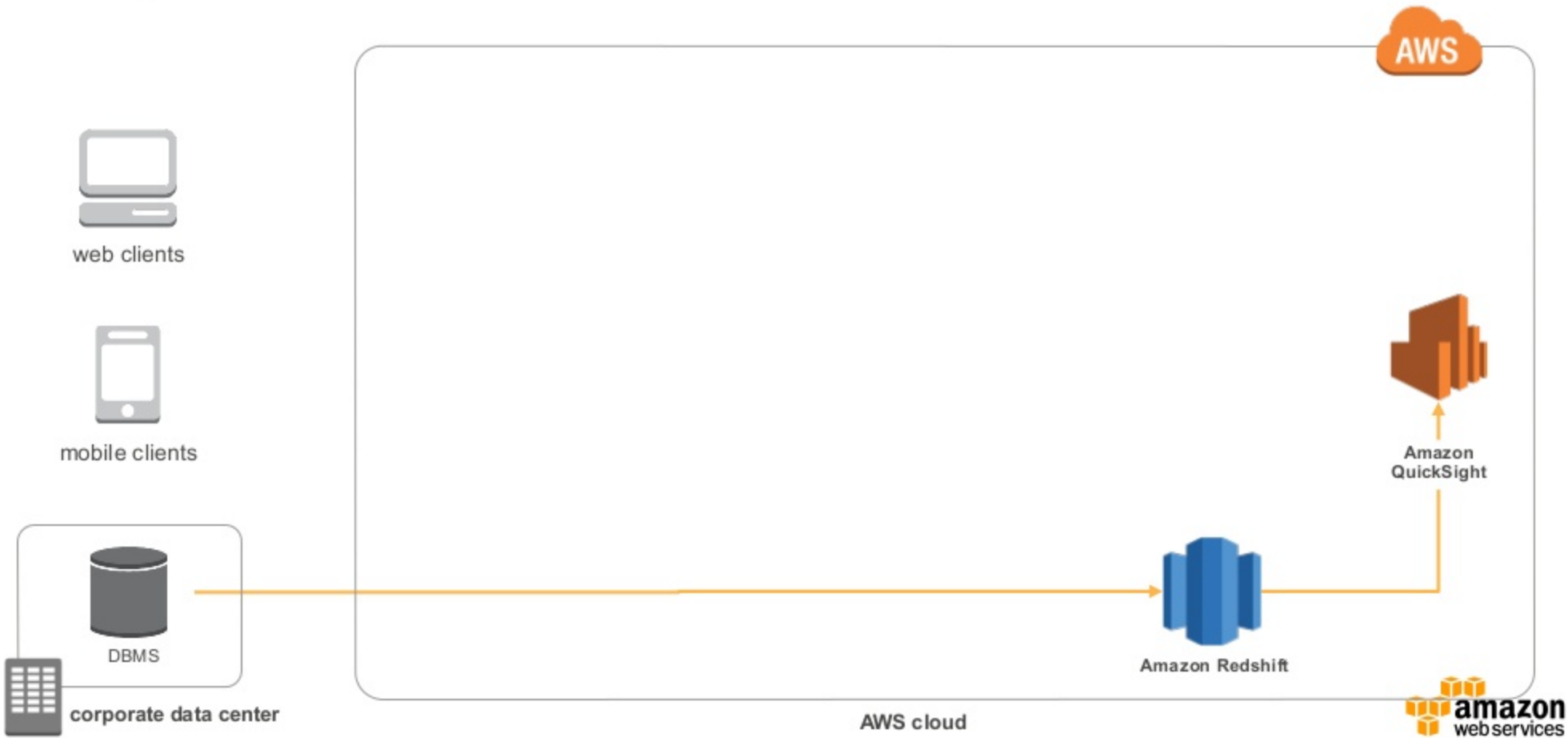
mobile clients



amazon web services

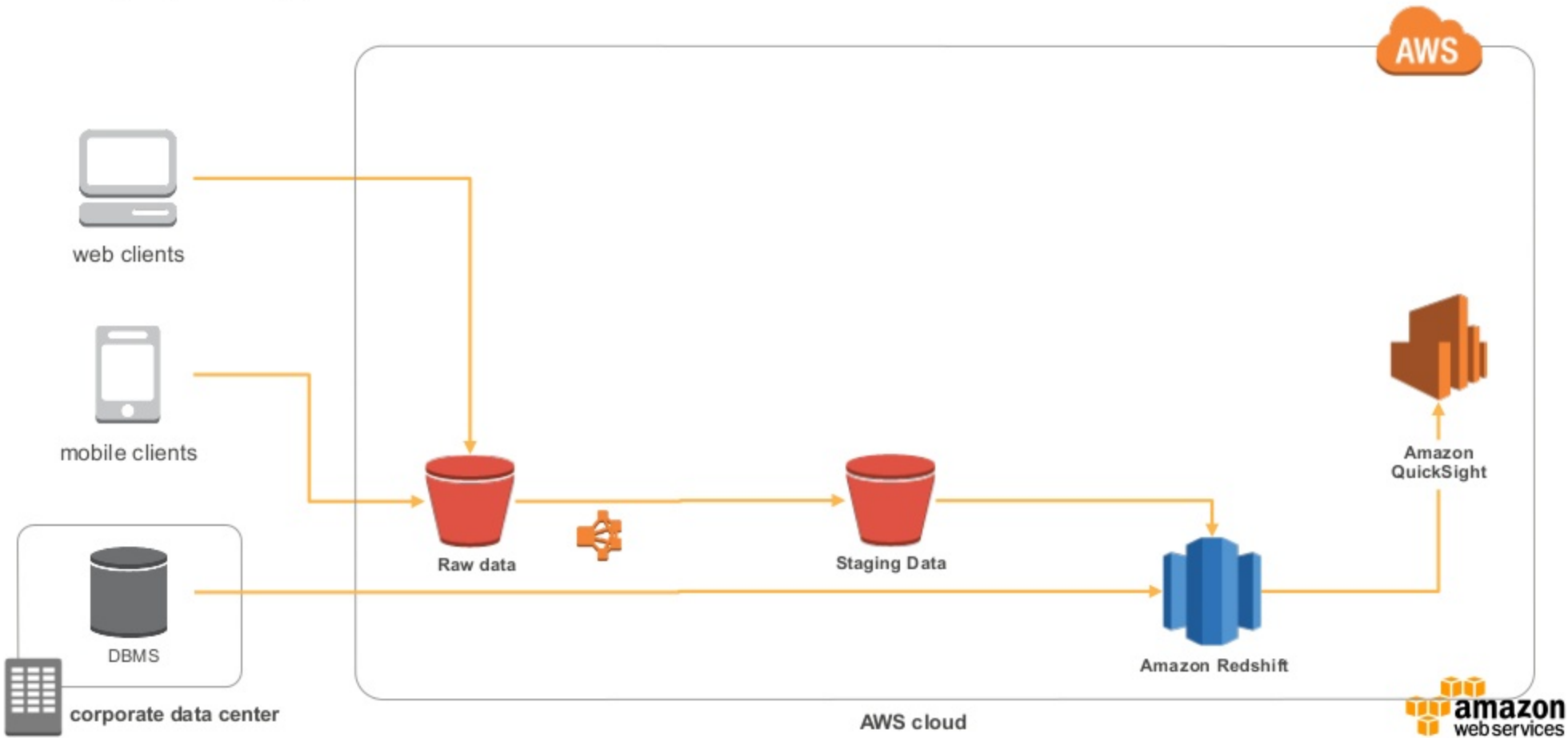
Building a Big Data Application

Adding a data warehouse



Building a Big Data Application

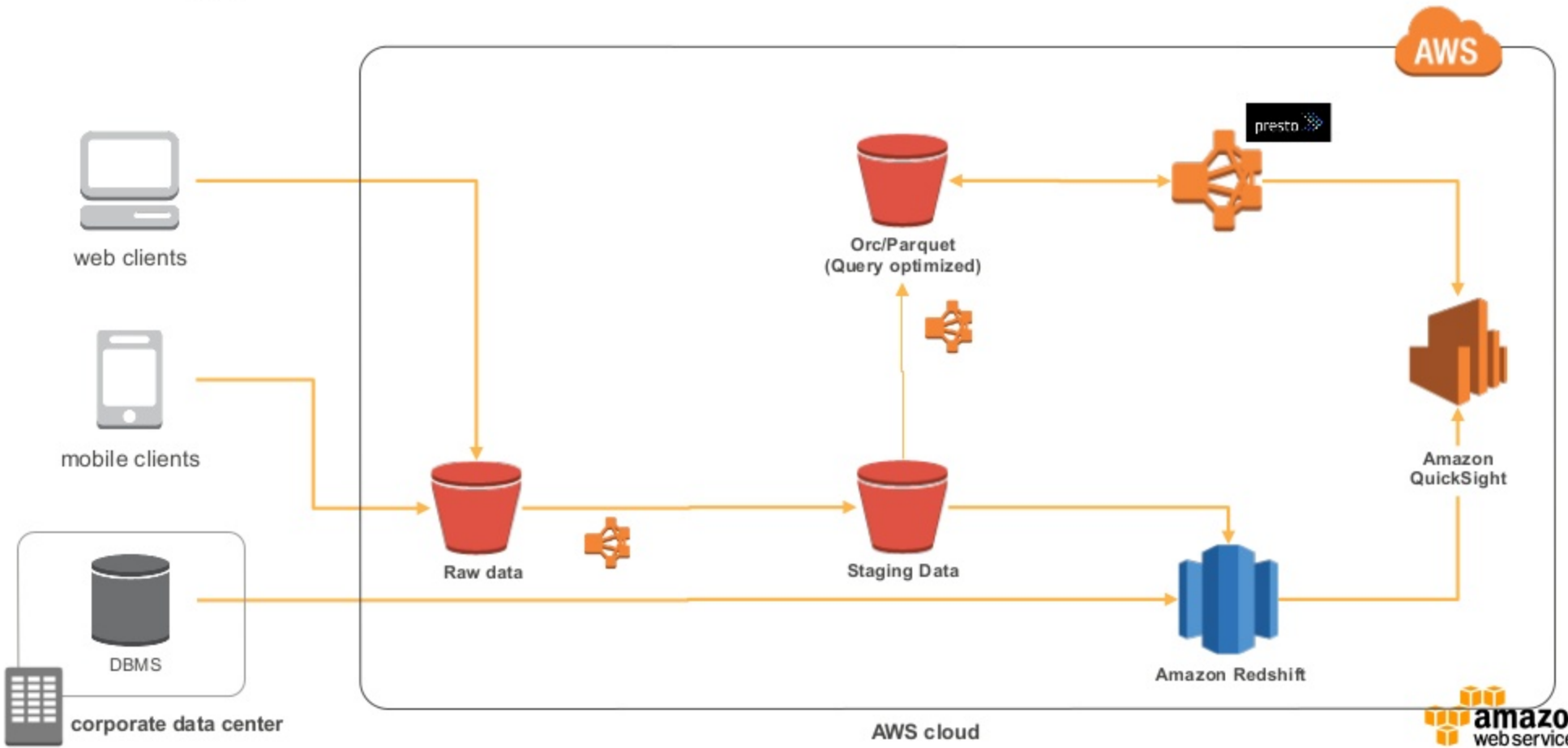
Bringing in Log Data



amazon web services

Building a Big Data Application

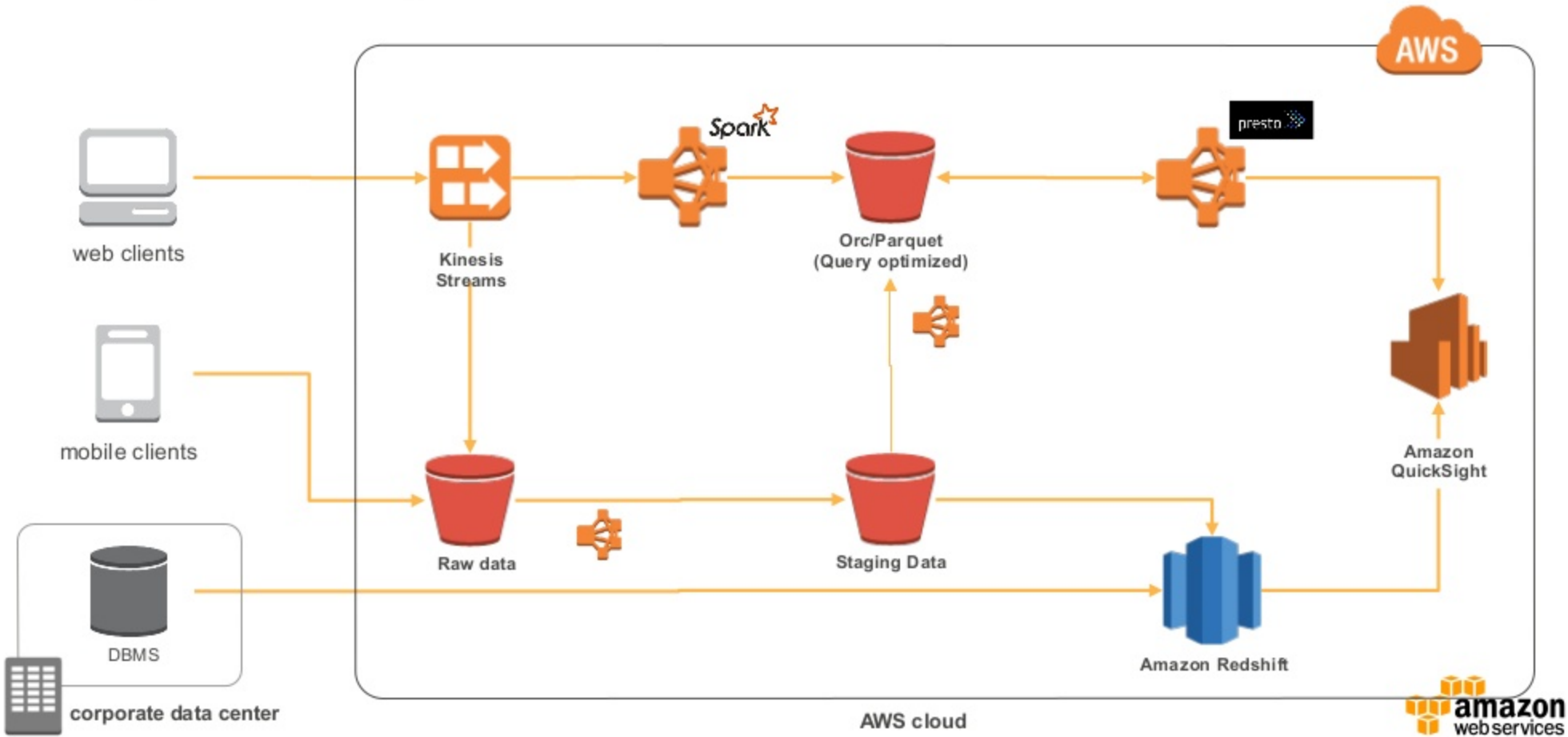
Extending your DW to S3



amazon web services

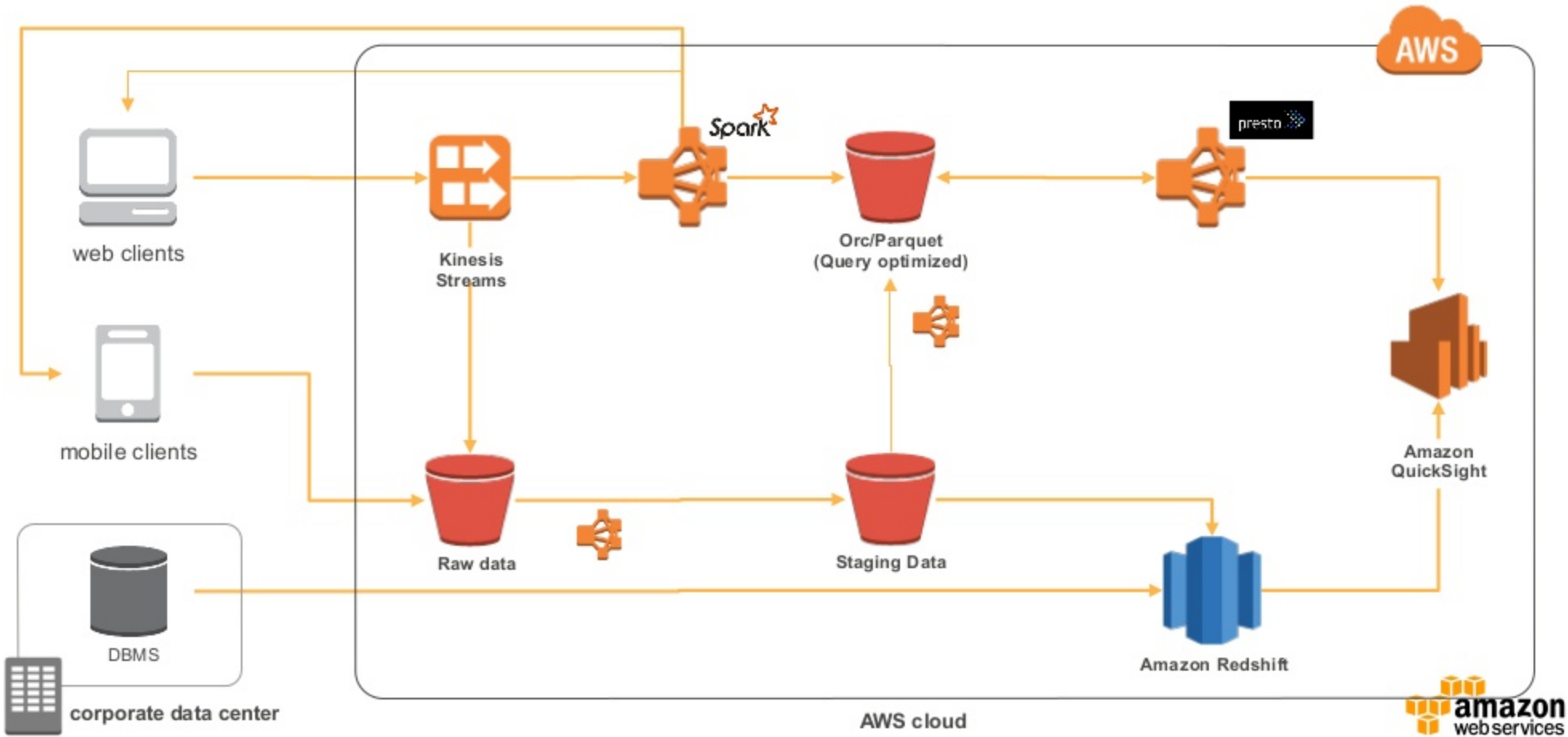
Building a Big Data Application

Adding a real-time layer



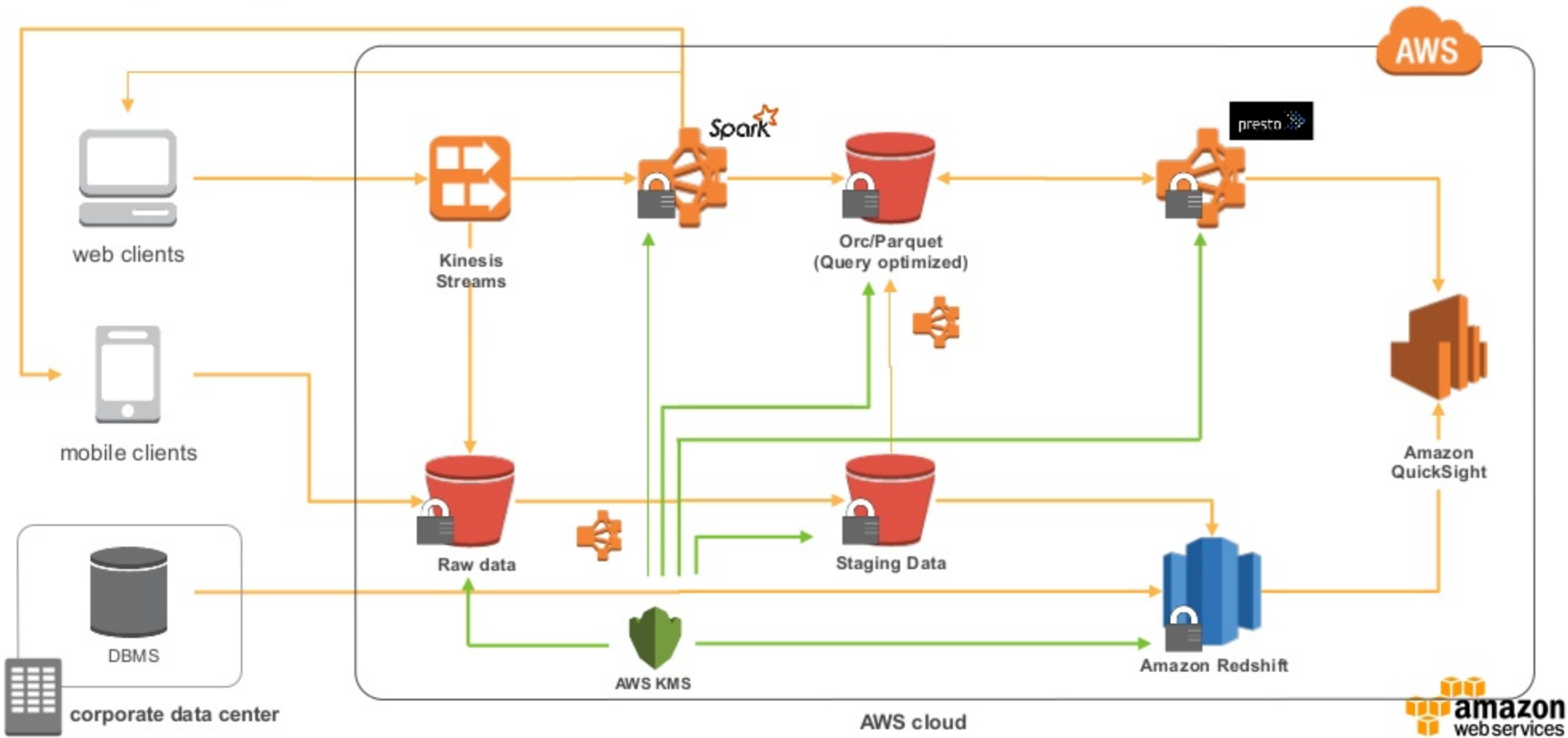
Building a Big Data Application

Adding predictive analytics



Building a Big Data Application

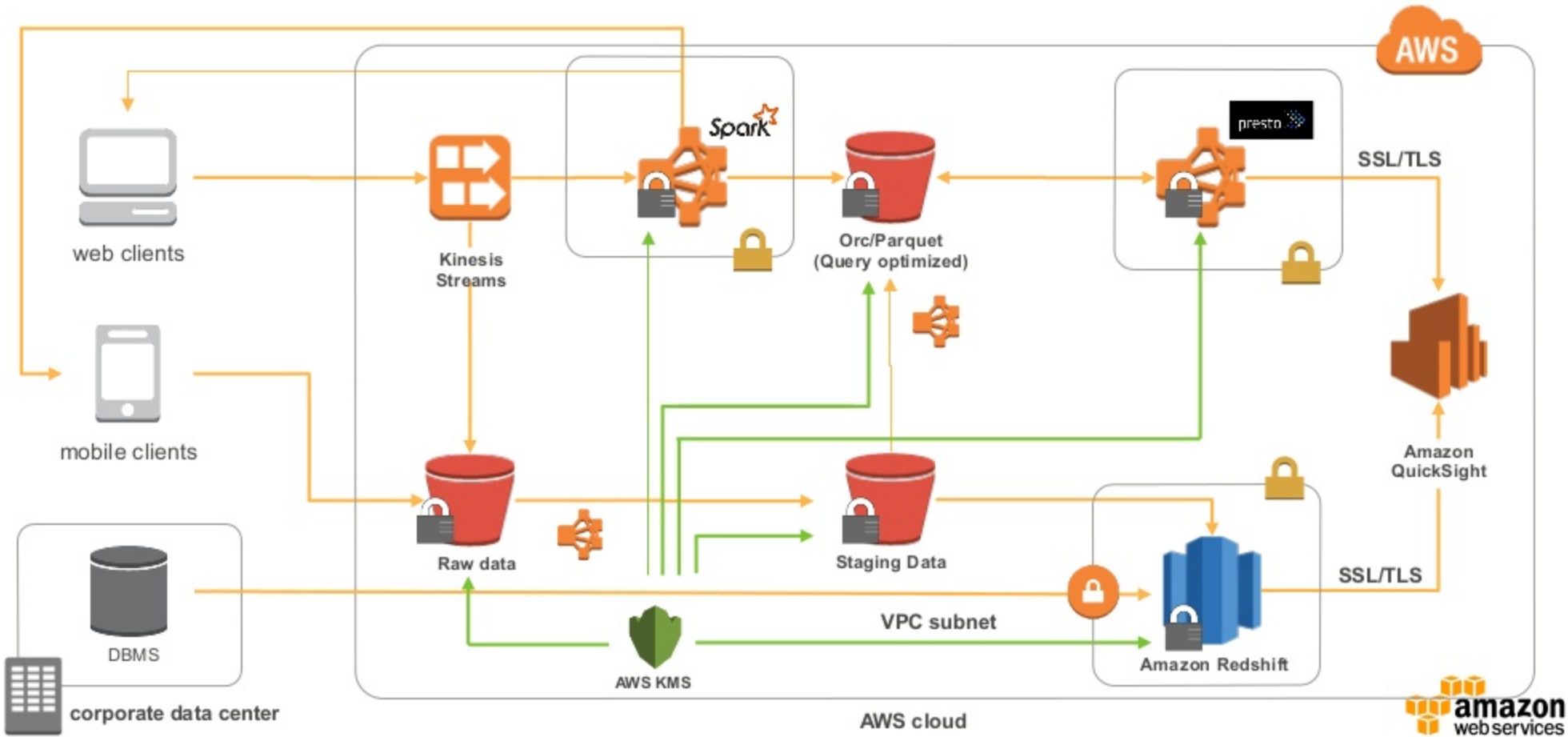
Adding encryption at rest with AWS KMS



amazon web services

Building a Big Data Application

Protecting Data in Transit & Adding Network Isolation

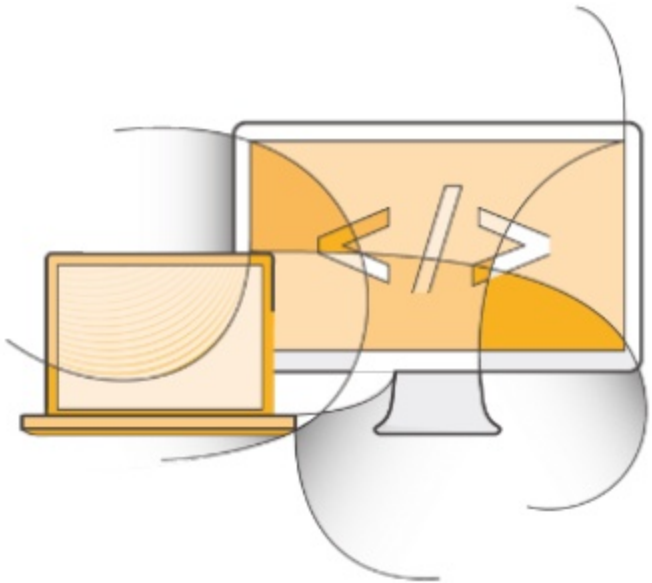


The Evolution of Big Data Processing

Descriptive	Real-time	Predictive
What happened before	What's happening now	Probability of "x" happening
Dashboards; Traditional query & reporting	Clickstream analysis; Ad bidding; streaming data	Inventory forecasting; Fraud detection; Recommendation engines
Amazon Redshift Amazon QuickSight Amazon EMR	Amazon Kinesis Amazon EC2 AWS Lambda Amazon DynamoDB Amazon EMR	Amazon Machine Learning Amazon EMR

amazon web services

PLACEHOLDER CALL TO ACTION



aws.amazon.com/big-data

AWS Courses

Big Data Technology Fundamentals FREE!
Overview of AWS big data solutions for architects or data scientists new to big data. (3 Hours | Online)

Big Data on AWS
How to use AWS services to process data with Hadoop & create big data environments (3 Days | Classroom)

Self-paced Online Labs

Big Data Quest
Learn at your own pace and practice working with AWS services for big data on qwikLABS. (3 Hours | Online)

amazon web services

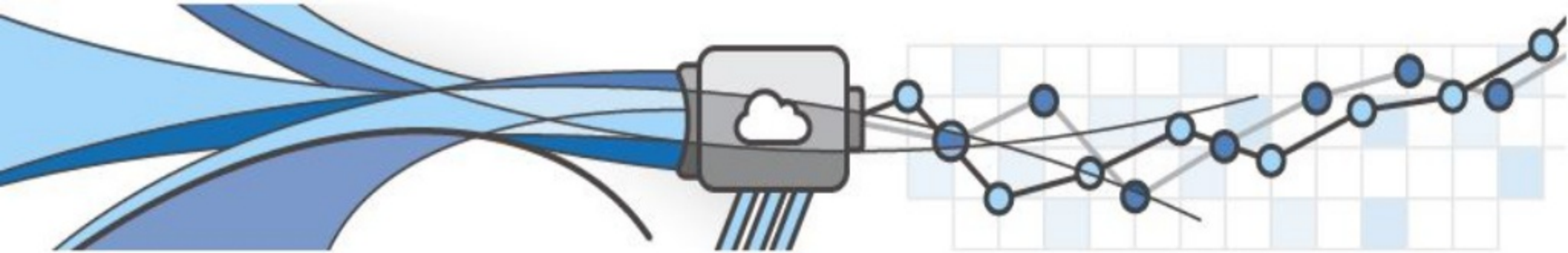
Next…

2:00 PM Best Practices on Real-time Streaming Analytics

3:00 PM Break

3:15 PM Getting started with Amazon Machine Learning

4:15 PM Building your first big data application on AWS



Real-time Streaming Data on AWS

Deep Dive & Best Practices Using Amazon Kinesis, Spark Streaming, AWS Lambda, and Amazon EMR

Steve Abraham
Solutions Architect, AWS

It’s All About the Pace

Batch Processing	Stream Processing
Hourly server logs	Real-time metrics
Weekly or monthly bills	Real-time spending alerts/caps
Daily web-site clickstream	Real-time clickstream analysis
Daily fraud reports	Real-time detection

Streaming Data Scenarios Across Verticals

Scenarios/ Verticals	Accelerated Ingest-Transform-Load	Continuous Metrics Generation	Responsive Data Analysis
Digital Ad Tech/ Marketing	Publisher, bidder data aggregation	Advertising metrics like coverage, yield, and conversion	User engagement with ads, optimized bid/buy engines
IoT	Sensor, device telemetry data ingestion	Operational metrics and dashboards	Device operational intelligence and alerts
Gaming	Online data aggregation, e.g., top 10 players	Massively multiplayer online game (MMOG) live dashboard	Leader board generation, player-skill match
Consumer Online	Clickstream analytics	Metrics like impressions and page views	Recommendation engines, proactive care

Customer Use Cases

SONOS

Sonos runs near real-time streaming analytics on device data logs from their connected hi-fi audio equipment.

Glu Mobile collects billions of gaming events data points from millions of user devices in real-time every single day.

HEARST

Analyzing 30TB+ clickstream data enabling real-time insights for Publishers.

Nordstorm

Nordstorm recommendation team built online stylist using Amazon Kinesis Streams and AWS Lambda.

Streaming Data Challenges: Variety & Velocity

- Streaming data comes in different types and formats
  - Metering records, logs and sensor data
  - JSON, CSV, TSV
- Can vary in size from a few bytes to kilobytes or megabytes
- High velocity and continuous

```
{
  "payerId": "Joe",
  "productCode": "AmazonS3",
  "clientProductCode": "AmazonS3",
  "usageType": "Bandwidth",
  "operation": "PUT",
  "value": "22490",
  "timestamp": "1216674828"
}
```

Metering Record

```
{
  127.0.0.1 user-
  identifier frank
  [10/Oct/
  2000:13:55:36
  -0700] "GET /
  apache_pb.gif
  HTTP/1.0" 200
  2326
}
```

Common Log Entry

```
{
  <165>1 2003-10-11T22:14:15.003Z
  mymachine.example.com evntslog -
  ID47 [exampleSDID@32473 iut="3"
  eventSource="Application"
  eventID="1011"]
  [examplePriority@32473
  class="high"]
}
```

Syslog Entry

```
{
   "SeattlePublicWa
   ter/Kinesis/123/
   Realtime" –
   412309129140
}
```

MQTT Record

Two Main Processing Patterns

Stream processing (real time)

- Real-time response to events in data streams

Examples:

- Proactively detect hardware errors in device logs
- Notify when inventory drops below a threshold
- Fraud detection

Micro-batching (near real time)

- Near real-time operations on small batches of events in data streams

Examples:

- Aggregate and archive events
- Monitor performance SLAs

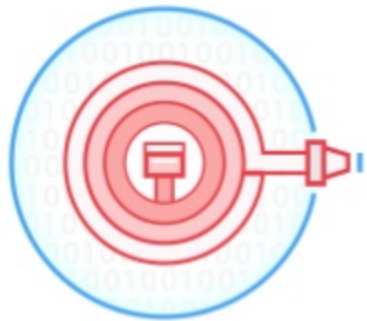
Amazon Kinesis Deep Dive

Amazon Kinesis: Streaming Data Made Easy

Services make it easy to capture, deliver and process streams on AWS

Amazon Kinesis Streams

- For Technical Developers
- Build your own custom applications that process or analyze streaming data



Amazon Kinesis Firehose

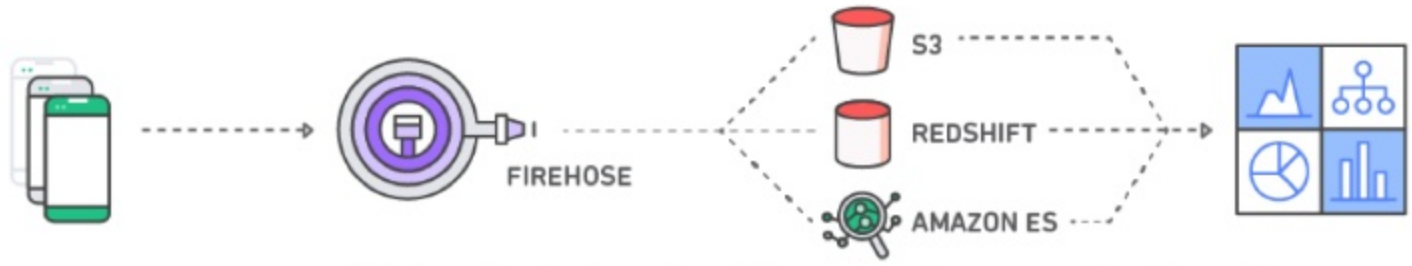
- For all developers, data scientists
- Easily load massive volumes of streaming data into S3, Amazon Redshift and Amazon Elasticsearch

Amazon Kinesis Analytics

- For all developers, data scientists
- Easily analyze data streams using standard SQL queries
- **Coming soon**

Amazon Kinesis Firehose

Load massive volumes of streaming data into Amazon S3, Amazon Redshift and Amazon Elasticsearch



Capture and submit streaming data to Firehose

Firehose loads streaming data continuously into S3, Amazon Redshift and Amazon Elasticsearch

Analyze streaming data using your favorite BI tools

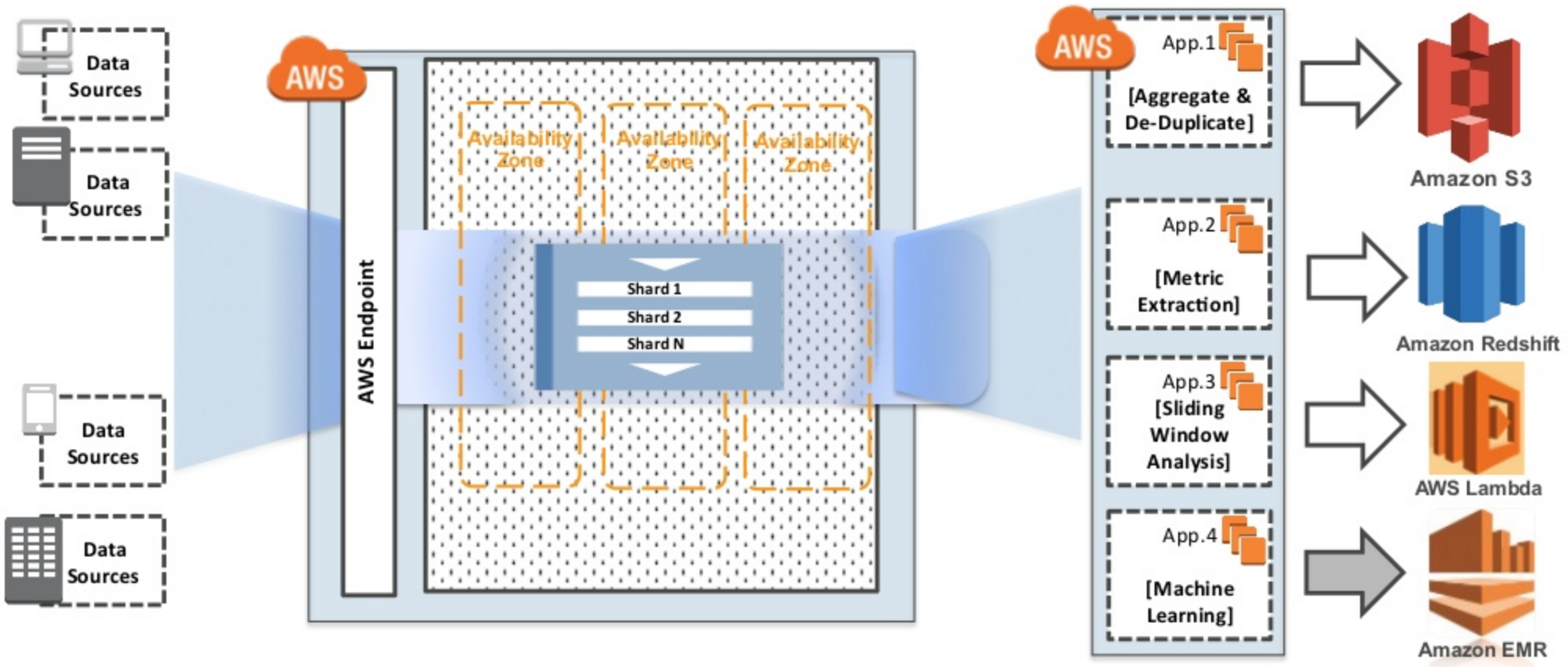
Zero administration: Capture and deliver streaming data into Amazon S3, Amazon Redshift and Amazon Elasticsearch **without writing an application or managing infrastructure.**

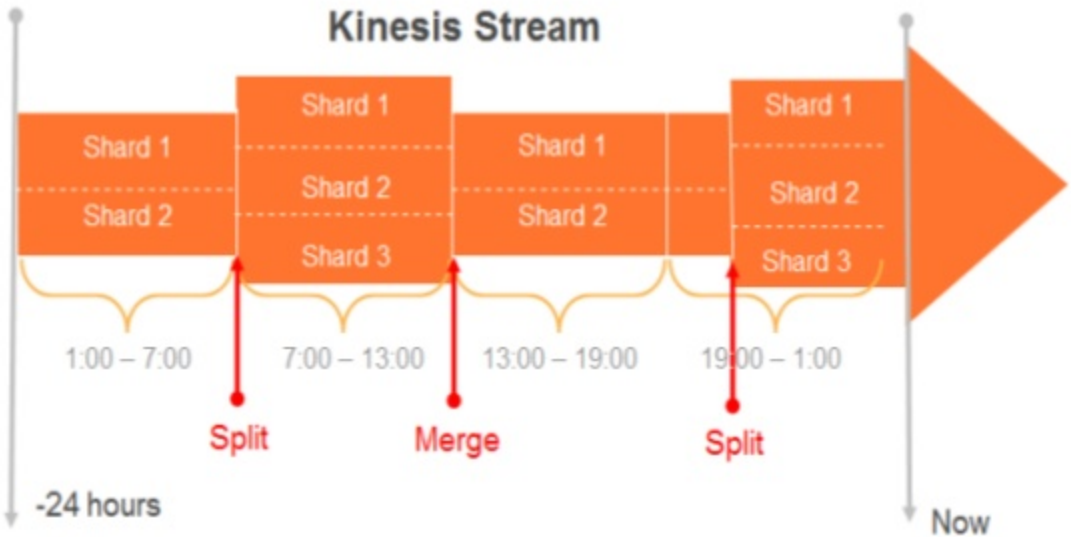
Direct-to-data store integration: **Batch, compress**, and **encrypt** streaming data for delivery into data destinations **in as little as 60 secs** using simple configurations.

Seamless elasticity: Seamlessly scales to match data throughput w/o intervention

Amazon Kinesis Streams

Managed service for real-time streaming



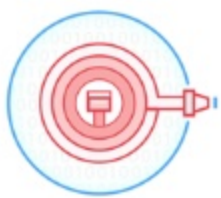


- Streams are made of **shards**
- Each shard ingests up to 1MB/sec, and 1000 records/sec
- Each shard emits up to 2 MB/sec
- All data is **stored for 24 hours by default**; storage can **be extended for up to 7 days**
- **Scale** Kinesis streams using scaling util
- **Replay** data inside of 24-hour window

Amazon Kinesis Firehose vs. Amazon Kinesis Streams

Amazon Kinesis Streams

Amazon Kinesis Streams is for use cases that require **custom processing**, per incoming record, with sub-1 second processing latency, and a choice of stream processing frameworks.



Amazon Kinesis Firehose

Amazon Kinesis Firehose is for use cases that require zero administration, ability to **use existing analytics tools based on Amazon S3, Amazon Redshift and Amazon Elasticsearch**, and a data latency of 60 seconds or higher.

Streaming Data Ingestion

Putting Data into Amazon Kinesis Streams

Determine your partition key strategy

- Managed buffer or streaming MapReduce job
- Ensure high cardinality for your shards

Provision adequate shards

- For ingress needs
- Egress needs for all consuming applications: if more than two simultaneous applications
- Include headroom for catching up with data in stream

Putting Data into Amazon Kinesis

Amazon Kinesis Agent – (supports pre-processing)

- http://docs.aws.amazon.com/firehose/latest/dev/writing-with-agents.html

Pre-batch before Puts for better efficiency

- Consider Flume, Fluentd as collectors/agents
- See https://github.com/awslabs/aws-fluent-plugin-kinesis

Make a tweak to your existing logging

- log4j appender option
- See https://github.com/awslabs/kinesis-log4j-appender

AWS IoT

- The Rules Engine can also route messages to AWS endpoints including Amazon Kinesis.

Putting Data into Amazon Kinesis Streams

Simple Put* interface to capture and store data in Streams

- A provisioned entity called a Stream composed of Shards
- Producers use a PUT call to store data in a Stream.
- Each record <= 1 MB. PutRecord or PutRecords
- A partition key is supplied by producer and used to distribute (MD5 hash) the PUTs across (hash key range) of Shards
- Unique Sequence# returned upon successful PUT call
- Approximate arrival timestamp affixed to each record

Amazon Kinesis Producer Library

- Writes to one or more Amazon Kinesis streams with automatic, configurable retry mechanism
- Collects records and uses PutRecords to write multiple records to multiple shards per request
- Aggregates user records to increase payload size and improve throughput
- Integrates seamlessly with KCL to de-aggregate batched records
- Use Amazon Kinesis Producer Library with AWS Lambda (**New!**)
- Submits Amazon CloudWatch metrics on your behalf to provide visibility into producer performance

Record Order and Multiple Shards

Unordered processing

- Randomize partition key to distribute events over many shards and use multiple workers

Exact order processing

- Control partition key to ensure events are grouped into the same shard and read by the same worker

Need both? Use global sequence number

Sample Code for Scaling Shards

```
java -cp
KinesisScalingUtils.jar-complete.jar
-Dstream-name=MyStream
-Dscaling-action=scaleUp
-Dcount=10
-Dregion=eu-west-1 ScalingClient
```

Options:

- **stream-name - The name of the stream to be scaled**
- **scaling-action - The action to be taken to scale. Must be one of "scaleUp", "scaleDown" or "resize"**
- **count - Number of shards by which to absolutely scale up or down, or resize**

See https://github.com/awslabs/amazon-kinesis-scaling-utils

Putting Data into Amazon Kinesis Streams

Determine your partition key strategy

Managed Buffer

- Care about a reliable, scalable way to capture data
- Defer all other aggregation to consumer
- Generate Random Partition Keys
- Ensure a high cardinality for Partition Keys with respect to shards, to spray evenly across available shards

Streaming Map-Reduce

- Streaming Map-Reduce: leverage partition keys as a natural way to aggregate data
- For e.g. Partition Keys per billing customer, per DeviceId, per stock symbol
- Design partition keys to scale
- Be aware of “hot partition keys or shards ”

Putting Data into Amazon Kinesis Streams

Provision adequate shards

- For ingress needs
- Egress needs for all consuming applications: if more than two simultaneous applications
- Include headroom for catching up with data in stream

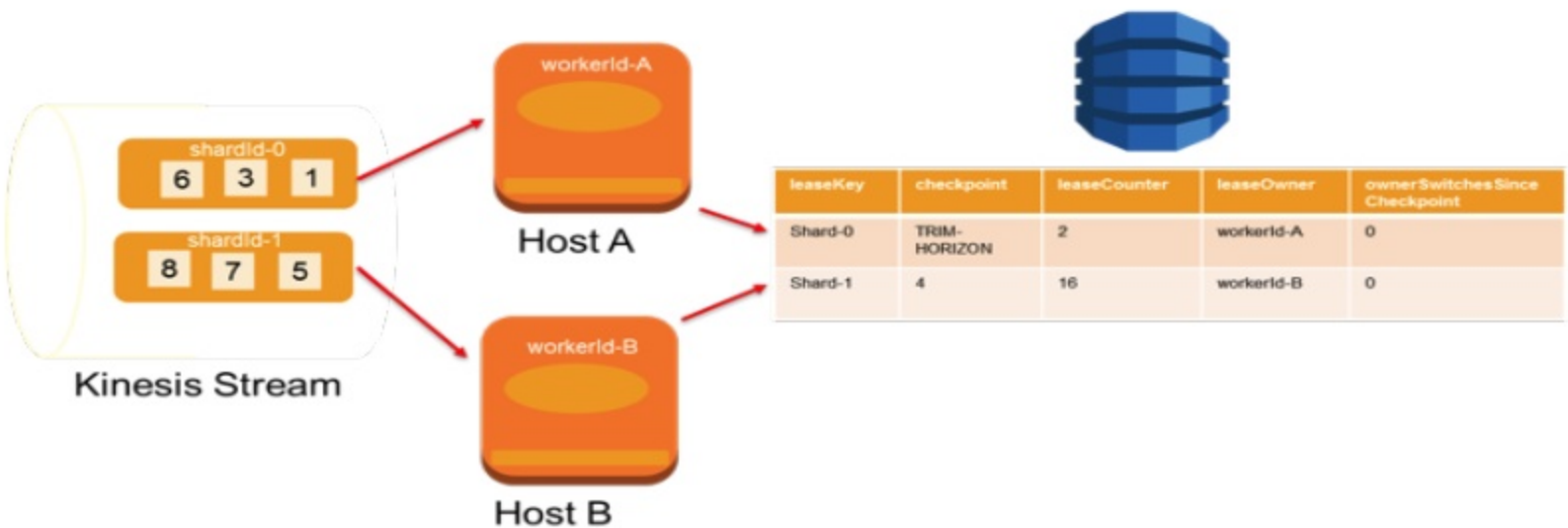
Amazon Kinesis Stream Processing

Amazon Kinesis Client Library

- Build Kinesis Applications with Kinesis Client Library (KCL)
- Open source client library available for Java, Ruby, Python, Node.JS dev
- Deploy on your EC2 instances
- KCL Application includes three components:
  1. **Record Processor Factory** – Creates the record processor
  2. **Record Processor** – Processor unit that processes data from a shard in Amazon Kinesis Streams
  3. **Worker** – Processing unit that maps to each application instance

State Management with Kinesis Client Library

- One record processor maps to one shard and processes data records from that shard
- One worker maps to one or more record processors
- Balances shard-worker associations when worker / instance counts change
- Balances shard-worker associations when shards split or merge



Other Options

- Third-party connectors (for example, Splunk)
- AWS IoT Platform
- **AWS Lambda**
- **Amazon EMR with Apache Spark, Pig or Hive**

Apache Spark and Amazon Kinesis Streams

Apache Spark is an **in-memory analytics** cluster using RDD for fast processing

Spark Streaming can read **directly** from an Amazon Kinesis stream

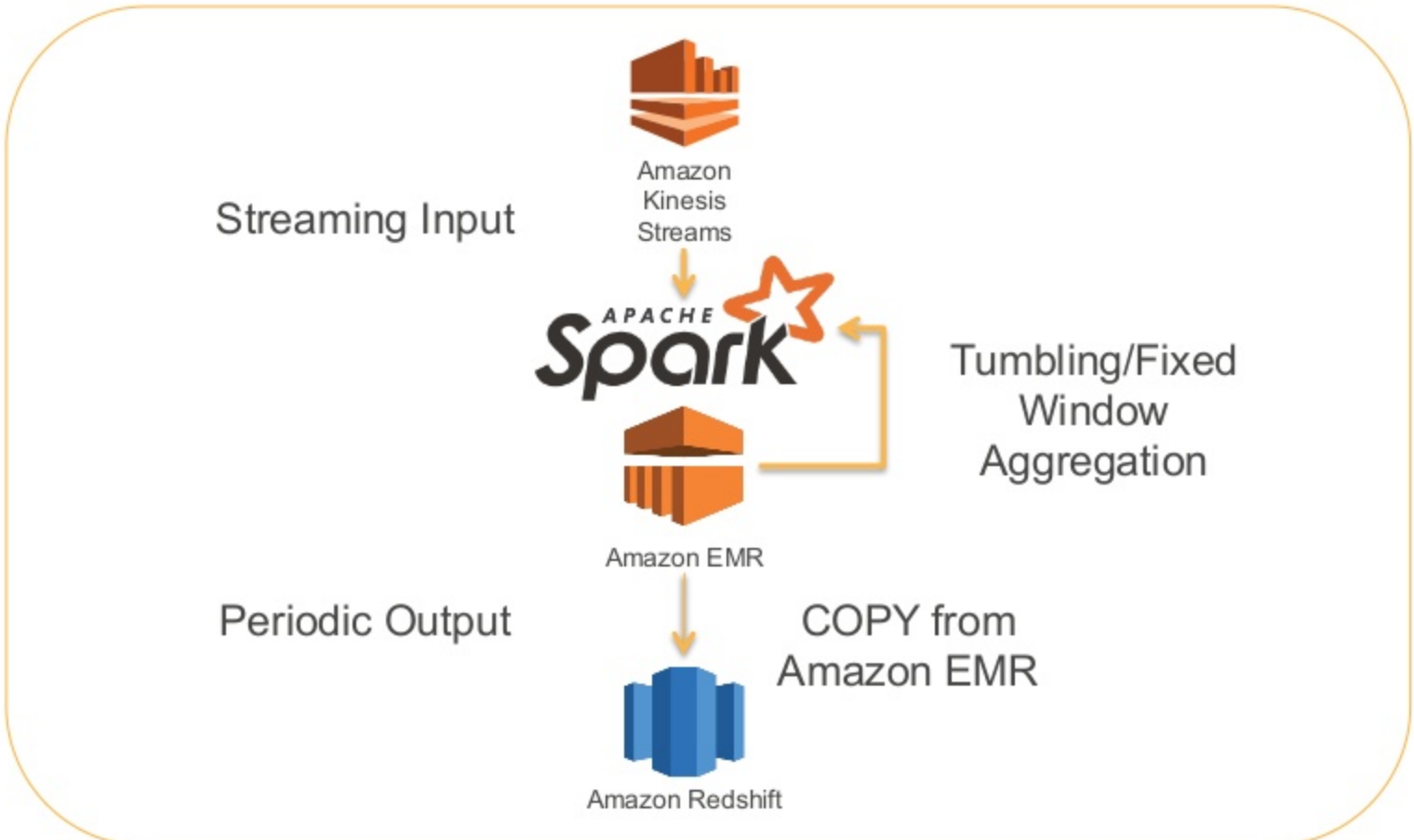
Amazon software license linking – Add ASL dependency to SBT/MAVEN project, `artifactId = spark-streaming-kinesis-asl_2.10`

Example: Counting tweets on a sliding window

```
KinesisUtils.createStream('twitter-stream')
    .filter(_.getText.contains("Open-Source"))
    .countByWindow(Seconds(5))
```

Common Integration Pattern with Amazon EMR

Tumbling Window Reporting



Using Spark Streaming with Amazon Kinesis Streams

1. **Use Spark 1.6+ with EMRFS consistent view option** – if you use Amazon S3 as storage for Spark checkpoint
2. **Amazon DynamoDB table name** – make sure there is only one instance of the application running with Spark Streaming
3. **Enable Spark-based checkpoints**
4. Number of Amazon Kinesis receivers is multiple of executors so they are load-balanced
5. Total processing time is less than the batch interval
6. Number of executors is the same as number of cores per executor
7. Spark Streaming uses default of 1 sec with KCL

Demo #1

Data Source → Collect / Store → Process → Consume

Sensors

- Simulated using EC2
- 20K events/sec

Amazon Kinesis Streams

- Scalable
- Managed
- Millisecond latency
- Highly available

Spark Streaming

- Scalable
- Managed
- Micro-batch
- Checkpoint in DynamoDB

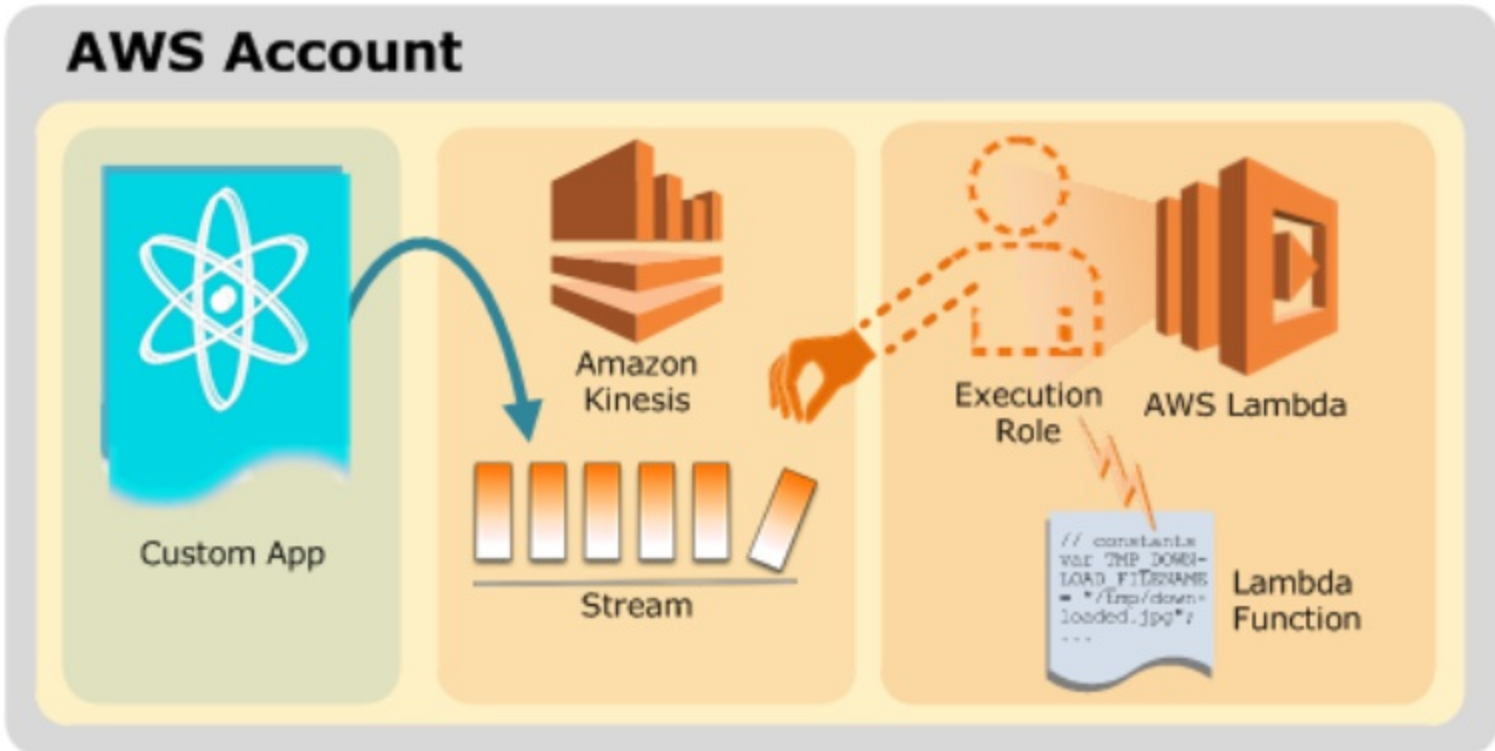
Apache Zeppelin

Device_Id, temperature, timestamp
1,65,2016-03-25 01:01:20
2,68,2016-03-25 01:01:20
3,67,2016-03-25 01:01:20
4,85,2016-03-25 01:01:20

Amazon DynamoDB

Click to add notes

Amazon Kinesis Streams with AWS Lambda



Build your Streaming Pipeline with Amazon Kinesis, AWS Lambda and Amazon EMR

Conclusion

- Amazon Kinesis offers: managed service to build applications, streaming data ingestion, and continuous processing
- Ingest aggregate data using Amazon Producer Library
- Process data using Amazon Connector Library and open source connectors
- Determine your partition key strategy
- Try out Amazon Kinesis at http://aws.amazon.com/kinesis/

Reference

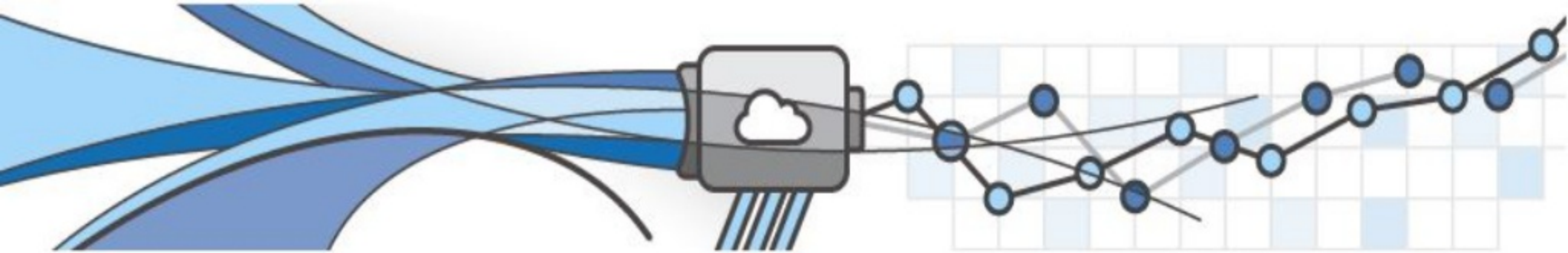
- **Technical documentation**
  - Amazon Kinesis Agent
  - Amazon Kinesis Streams and Spark Streaming
  - Amazon Kinesis Producer Library Best Practice
  - Amazon Kinesis Firehose and AWS Lambda
  - Building Near Real-Time Discovery Platform with Amazon Kinesis
- **Public case studies**
  - Glu mobile – Real-Time Analytics
  - Hearst Publishing – Clickstream Analytics
  - How Sonos Leverages Amazon Kinesis
  - Nordstorm Online Stylist

Next…

3:00 PM Break

3:15 PM Getting started with Amazon Machine Learning

4:15 PM Building your first big data application on AWS



Getting Started with Amazon Machine Learning

Fabio Silva

Solution Architect, AWS

Agenda

- Machine learning and the data ecosystem
- Smart applications by example (and counterexample)
- Amazon Machine Learning (Amazon ML) features and benefits
- Developing with Amazon ML
- Q&A

Three types of data-driven development

Retrospective
analysis and
reporting

Amazon Redshift,
Amazon RDS
Amazon S3
Amazon EMR

Three types of data-driven development

Retrospective
analysis and
reporting

Amazon Redshift,
Amazon RDS
Amazon S3
Amazon EMR

Here-and-now
real-time processing
and dashboards

Amazon Kinesis
Amazon EC2
AWS Lambda

Three types of data-driven development

Retrospective
analysis and
reporting

Amazon Redshift,
Amazon RDS
Amazon S3
Amazon EMR

Here-and-now
real-time processing
and dashboards

Amazon Kinesis
Amazon EC2
AWS Lambda

Predictions
to enable smart
applications

Machine learning and smart applications

Machine learning is the technology that automatically finds patterns in your data and uses them to make predictions for new data points as they become available.

Machine learning and smart applications

Machine learning is the technology that automatically finds patterns in your data and uses them to make predictions for new data points as they become available.

Your data + machine learning = smart applications

Smart applications by example

Based on what you know about the **user**:

Will they use your product?

Smart applications by example

Based on what you know about the **user**:

Will they use your product?

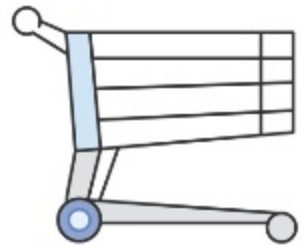
Based on what you know about an **order**:

Is this order fraudulent?

Smart applications by example

Based on what you know about the **user**:

Will they use your product?



Based on what you know about an **order**:

Is this order fraudulent?

Based on what you know about a **news article**:

What other articles are interesting?

And a few more examples…

Fraud detection	Detecting fraudulent transactions, filtering spam emails, flagging suspicious reviews,…
Personalization	Recommending content, predictive content loading, improving user experience,…
Targeted marketing	Matching customers and offers, choosing marketing campaigns, cross-selling and up-selling,…
Content classification	Categorizing documents, matching hiring managers and resumes,…
Churn prediction	Finding customers who are likely to stop using the service, upgrade targeting,…
Customer support	Predictive routing of customer emails, social media listening,…

Smart applications by *counterexample*

```
Dear Alex,

This awesome quadcopter is on sale
for just $49.99!
```

Smart applications by *counterexample*

```
SELECT  c.ID
FROM    customers c
        LEFT JOIN orders o
            ON c.ID = o.customer
GROUP   BY c.ID
HAVING  o.date > GETDATE() – 30
```

We can start by sending the offer to all customers who placed an order in the last 30 days

Smart applications by *counterexample*

```
SELECT   c.ID
FROM     customers c
         LEFT JOIN orders o
             ON c.ID = o.customer
GROUP    BY c.ID
HAVING   O.CATEGORY = 'TOYS'
   AND   o.date > GETDATE() – 30
```

…let's narrow it down to just customers who bought toys

Smart applications by *counterexample*

```
SELECT  c.ID
FROM    customers c
        LEFT JOIN orders o
            ON c.ID = o.customer
        LEFT JOIN PRODUCTS P
            ON P.ID = O.PRODUCT
GROUP   BY c.ID
HAVING  o.category = 'toys'
   AND ((P.DESCRIPTION LIKE '%HELICOPTER%'
         AND O.DATE > GETDATE() - 60)
     OR (COUNT(*) > 2
         AND  SUM(o.price) > 200
         AND  o.date > GETDATE() – 30)
       )
```

…and expand the query to customers who purchased other toy helicopters recently, or made several expensive toy purchases

Smart applications by *counterexample*

```
SELECT  c.ID
FROM    customers c
        LEFT JOIN orders o
            ON c.ID = o.customer
        LEFT JOIN products p
            ON p.ID = o.product
GROUP   BY c.ID
HAVING  o.category = 'toys'
   AND ((p.description LIKE '%COPTER%'
         AND o.date > GETDATE() - 60)
     OR (COUNT(*) > 2
         AND  SUM(o.price) > 200
         AND  o.date > GETDATE() – 30)
       )
```

…but what about quadcopters?

Smart applications by *counterexample*

```
SELECT  c.ID
FROM    customers c
        LEFT JOIN orders o
            ON c.ID = o.customer
        LEFT JOIN products p
            ON p.ID = o.product
GROUP   BY c.ID
HAVING  o.category = 'toys'
   AND ((p.description LIKE '%copter%'
         AND o.date > GETDATE() - 120)
     OR (COUNT(*) > 2
         AND  SUM(o.price) > 200
         AND  o.date > GETDATE() – 30)
       )
```

…maybe we should go back further in time

Smart applications by *counterexample*

```
SELECT  c.ID
FROM    customers c
        LEFT JOIN orders o
            ON c.ID = o.customer
        LEFT JOIN products p
            ON p.ID = o.product
GROUP   BY c.ID
HAVING  o.category = 'toys'
   AND ((p.description LIKE '%copter%'
         AND o.date > GETDATE() - 120)
     OR (COUNT(*) > 2
         AND  SUM(o.price) > 200
         AND  o.date > GETDATE() – 40)
       )
```

…tweak the query more

Smart applications by *counterexample*

```
SELECT  c.ID
FROM    customers c
        LEFT JOIN orders o
            ON c.ID = o.customer
        LEFT JOIN products p
            ON p.ID = o.product
GROUP   BY c.ID
HAVING  o.category = 'toys'
   AND ((p.description LIKE '%copter%'
         AND o.date > GETDATE() - 120)
     OR (COUNT(*) > 2
         AND  SUM(o.price) > 150
         AND  o.date > GETDATE() – 40)
       )
```

…again

Smart applications by *counterexample*

```
SELECT  c.ID
FROM    customers c
        LEFT JOIN orders o
            ON c.ID = o.customer
        LEFT JOIN products p
            ON p.ID = o.product
GROUP   BY c.ID
HAVING  o.category = 'toys'
   AND ((p.description LIKE '%copter%'
         AND o.date > GETDATE() - 90)
     OR (COUNT(*) > 2
         AND  SUM(o.price) > 150
         AND  o.date > GETDATE() – 40)
       )
```

…and again

Smart applications by *counterexample*

Use machine learning technology to **learn** your business rules from data!

Why aren't there *more* smart applications?

1. Machine learning expertise is **rare**.

2. Building and scaling machine learning technology is **hard**.

3. Closing the gap between models and applications is **time-consuming and expensive**.

Building smart applications today

Expertise	Technology	Operationalization
Limited supply of data scientists	Many choices, few mainstays	Complex and error-prone data workflows
Expensive to hire or outsource	Difficult to use and scale	Custom platforms and APIs
	Many moving pieces lead to custom solutions *every time*	Reinventing the model lifecycle management wheel

What if there were a better way?

Introducing Amazon Machine Learning

Easy-to-use, managed machine learning service built for developers

Robust, powerful machine learning technology based on Amazon's internal systems

Create models using your data already stored in the AWS cloud

Deploy models to production in seconds

Easy-to-use and developer-friendly

Use the intuitive, powerful service console to build and explore your initial models

- Data retrieval
- Model training, quality evaluation, fine-tuning
- Deployment and management

Automate model lifecycle with fully featured APIs and SDKs

- Java, Python, .NET, JavaScript, Ruby, PHP

Easily create smart iOS and Android applications with AWS Mobile SDK

Powerful machine learning technology

Based on Amazon's battle-hardened internal systems

Not just the algorithms:

- Smart data transformations
- Input data and model quality alerts
- Built-in industry best practices

Grows with your needs

- Train on up to 100 GB of data
- Generate billions of predictions
- Obtain predictions in batches or real-time

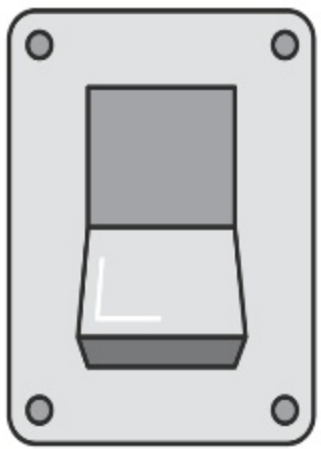
Integrated with the AWS data ecosystem

Access data that is stored in Amazon S3, Amazon Redshift, or MySQL databases in Amazon RDS

Output predictions to Amazon S3 for easy integration with your data flows

Use AWS Identity and Access Management (IAM) for fine-grained data access permission policies

Fully-managed model and prediction services



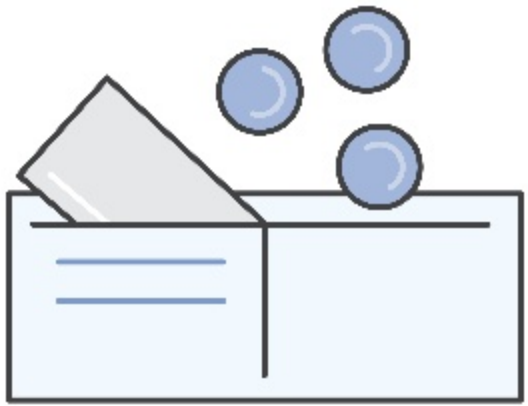
End-to-end service, with no servers to provision and manage

One-click production model deployment

Programmatically query model metadata to enable automatic retraining workflows

Monitor prediction usage patterns with Amazon CloudWatch metrics

Pay-as-you-go and inexpensive



Data analysis, model training, and evaluation: **$0.42/instance hour**

Batch predictions: **$0.10/1000**

Real-time predictions: **$0.10/1000**

+ hourly capacity reservation charge

Three supported types of predictions

Binary classification

Predict the answer to a Yes/No question

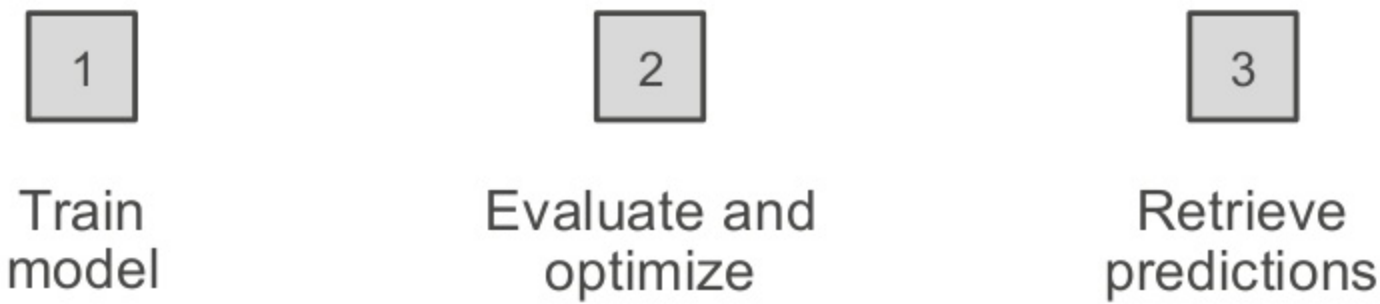
Multiclass classification

Predict the correct category from a list

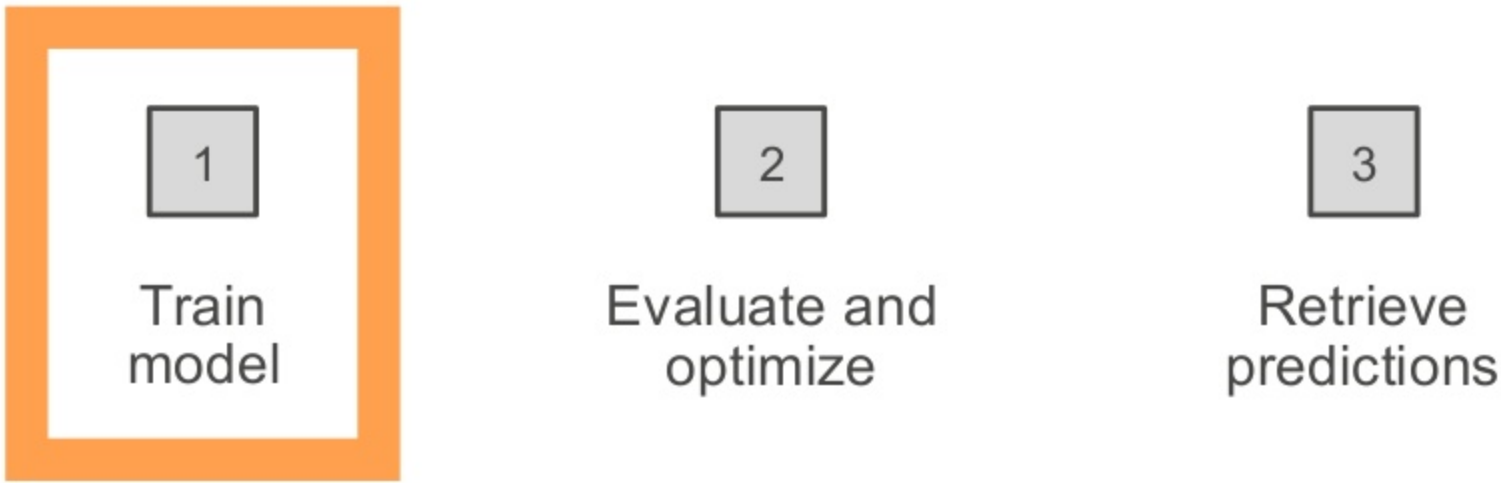
Regression

Predict the value of a numeric variable

Building smart applications with Amazon ML

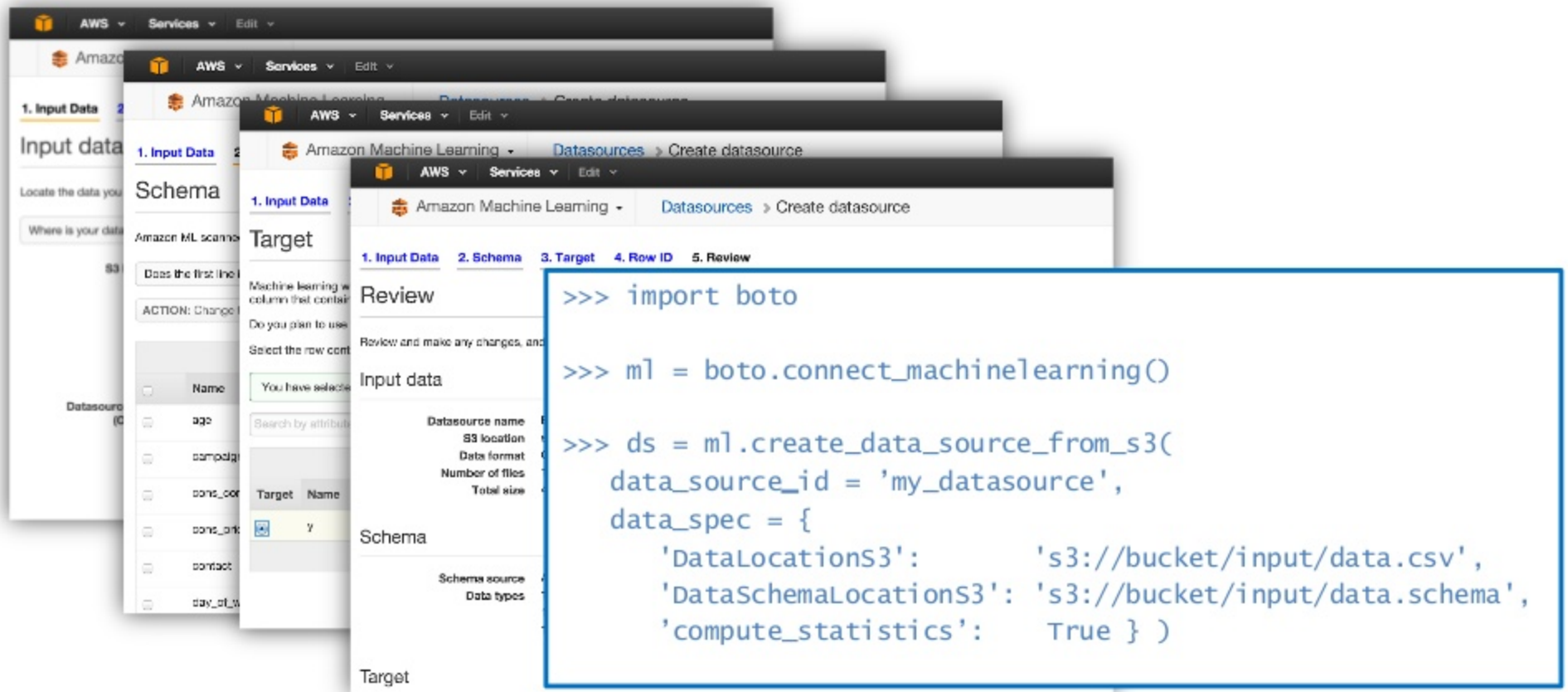


Building smart applications with Amazon ML

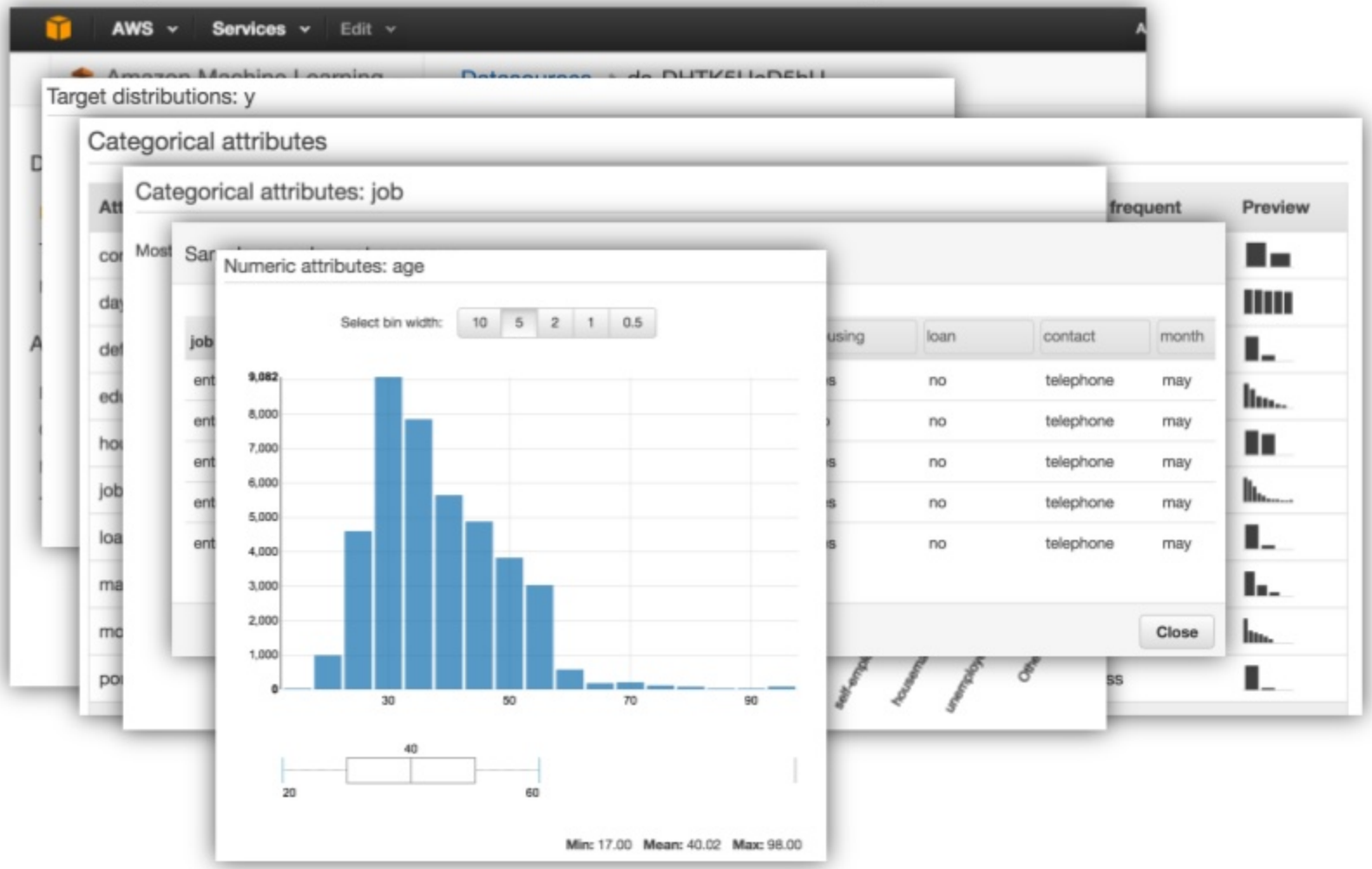


- Create a datasource object pointing to your data
- Explore and understand your data
- Transform data and train your model

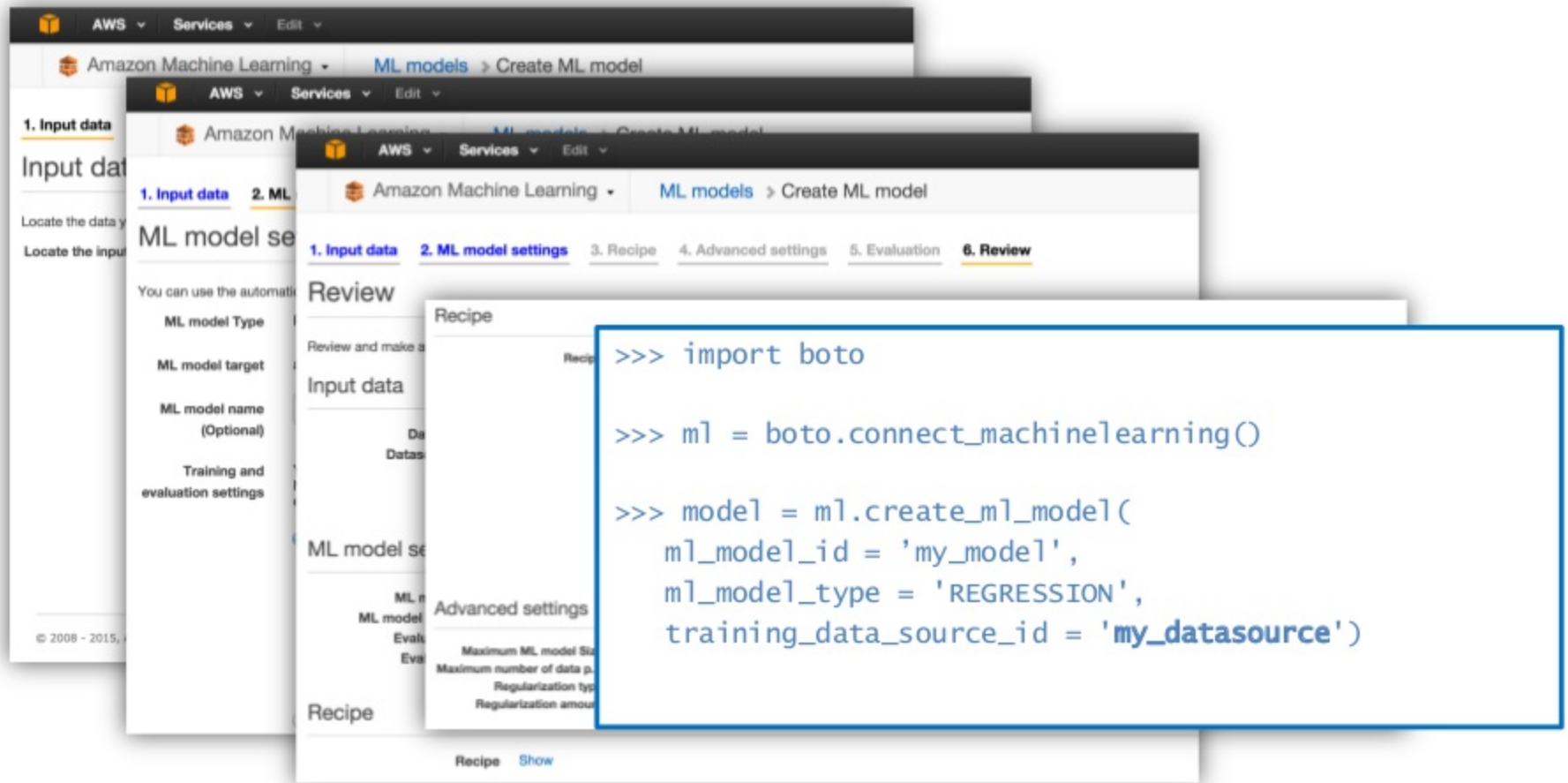
Create a datasource object



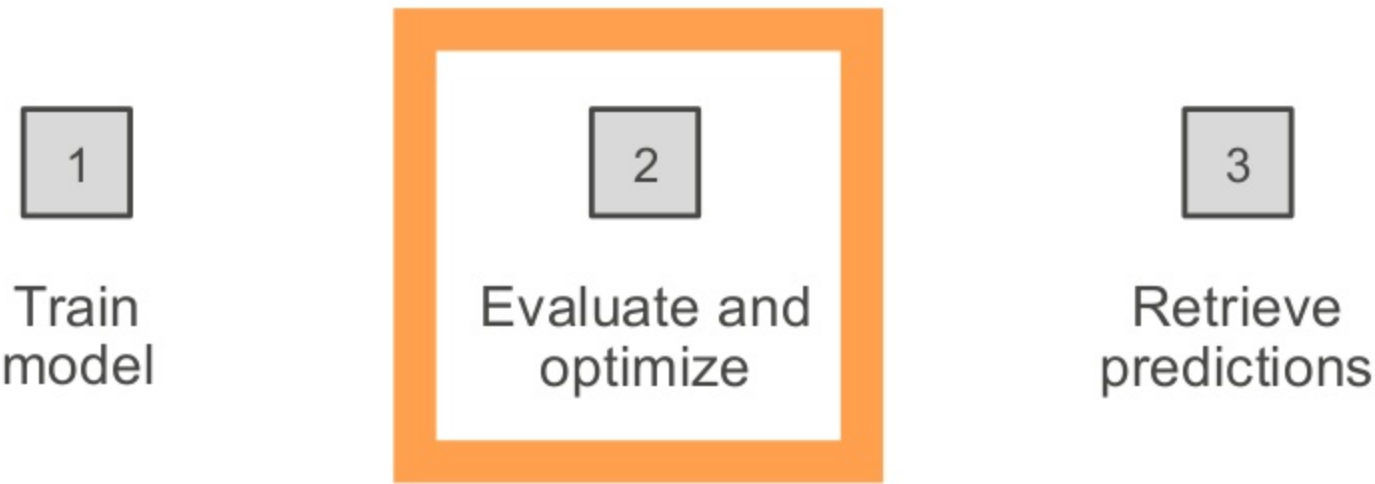
Explore and understand your data



Train your model



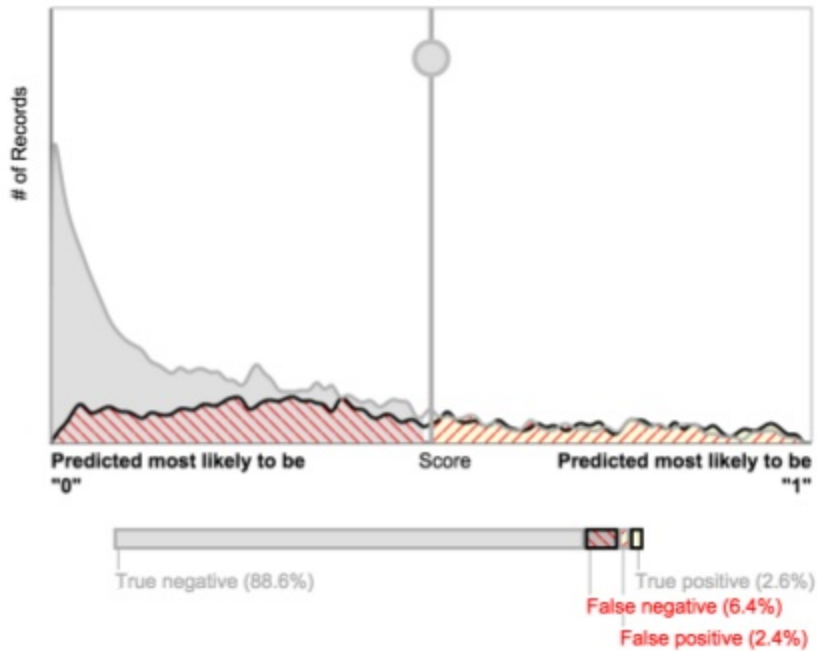
Building smart applications with Amazon ML



- Measure and understand model quality
- Adjust model interpretation

Explore model quality

Fine-tune model interpretation



Trade-off based on score threshold 0.5

Reset score threshold (0.5)

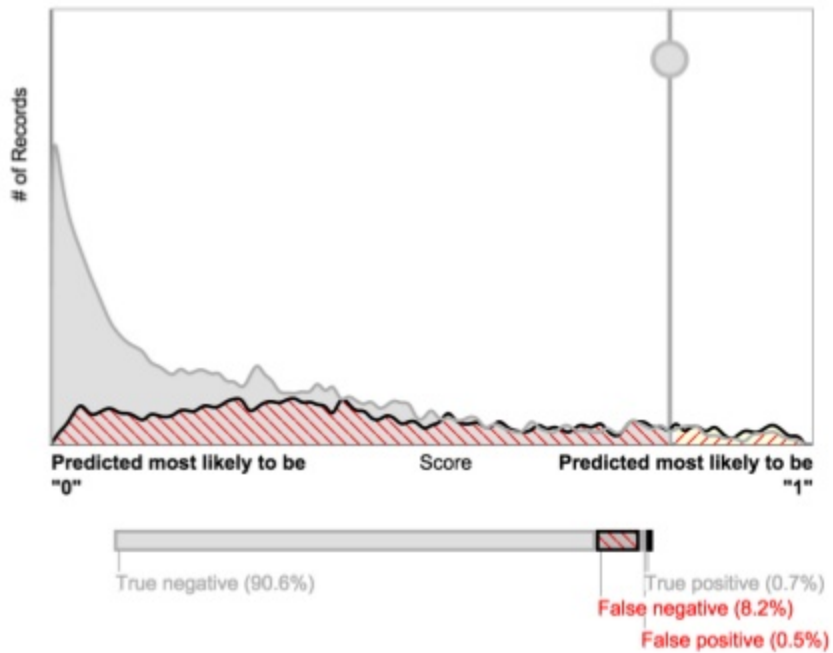
- **91% are correct**
  607 true positive
  20,933 true negative
- **9% are errors**
  567 false positive
  1,507 false negative

- 5% of the records are predicted as "1"
- 95% of the records are predicted as "0"

Save score threshold at 0.50

› Advance metrics

Fine-tune model interpretation



Trade-off based on score threshold 0.81

Reset score threshold (0.5)

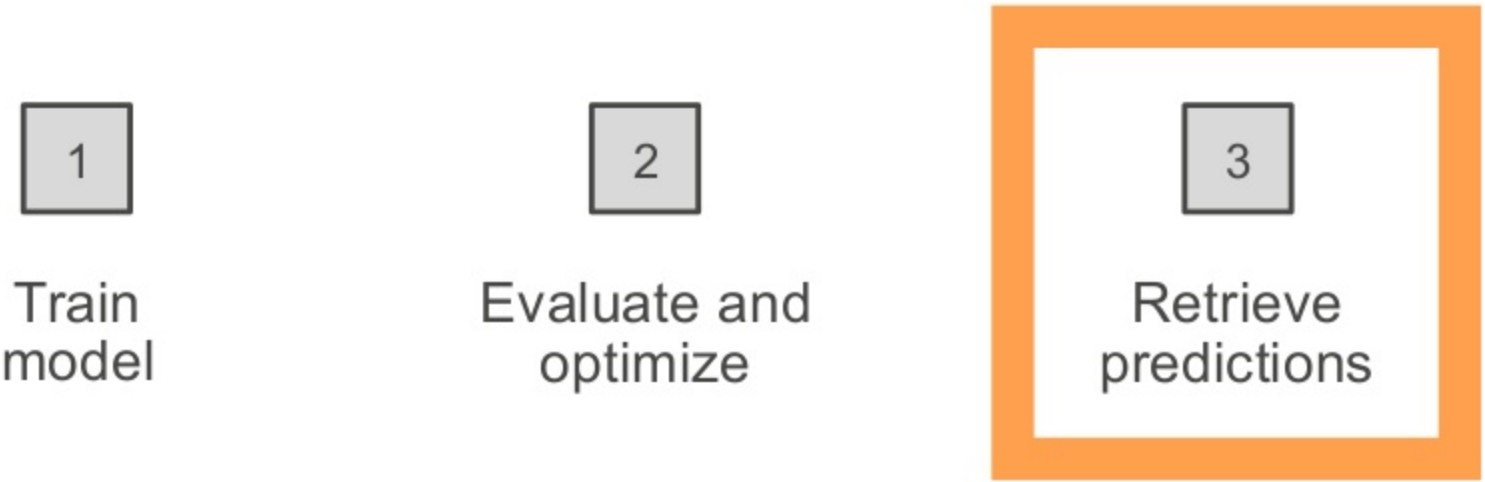
- **91% are correct**
  166 true positive
  21,385 true negative
- **9% are errors**
  115 false positive
  1,948 false negative

- 1% of the records are predicted as "1"
- 99% of the records are predicted as "0"

Save score threshold at 0.81

› Advance metrics

Building smart applications with Amazon ML



- Batch predictions
- Real-time predictions

Batch predictions

Asynchronous, large-volume prediction generation

Request through service console or API

Best for applications that deal with batches of data records

```
>>> import boto

>>> ml = boto.connect_machinelearning()

>>> model = ml.create_batch_prediction(
   batch_prediction_id = 'my_batch_prediction',
   batch_prediction_data_source_id = 'my_datasource',
   ml_model_id = 'my_model',
   output_uri = 's3://examplebucket/output/')
```

```
bestAnswer,score
0,3.93914E-1
0,1.654963E-2
1,0.832306
0,2.143189E-2
0,9.23001E-3
0,0.714461
0,9.772378E-3
1,0.525307
1,0.710729
```

Real-time predictions

Synchronous, low-latency, high-throughput prediction generation

Request through service API, server, or mobile SDKs

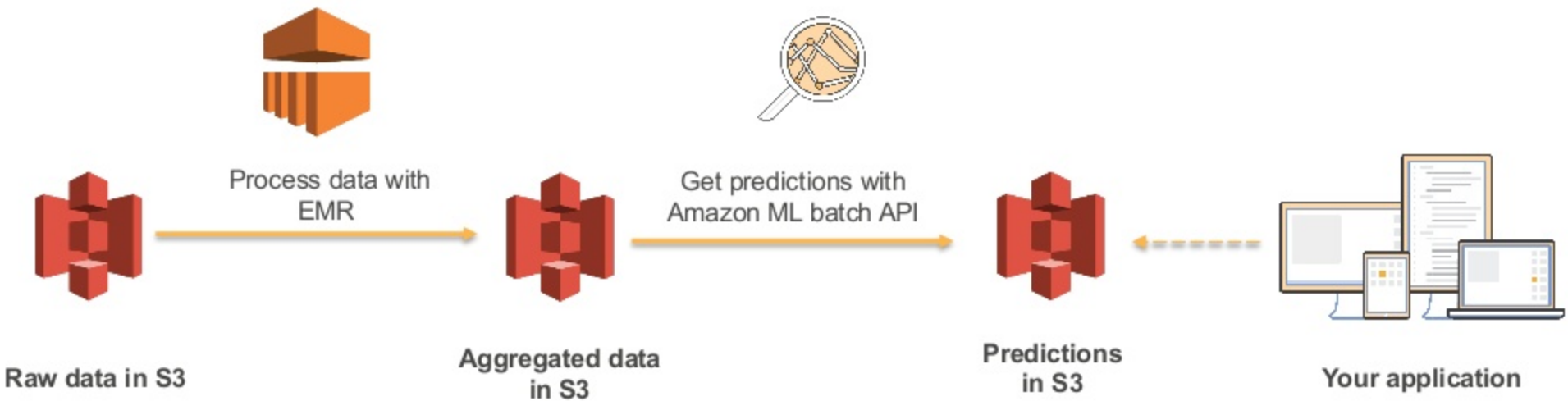
Best for interaction applications that deal with individual data records

```
>>> import boto

>>> ml = boto.connect_machinelearning()

>>> ml.predict(
    ml_model_id = 'my_model',
    predict_endpoint = 'example_endpoint',
    record = {'key1':'value1', 'key2':'value2'})
```

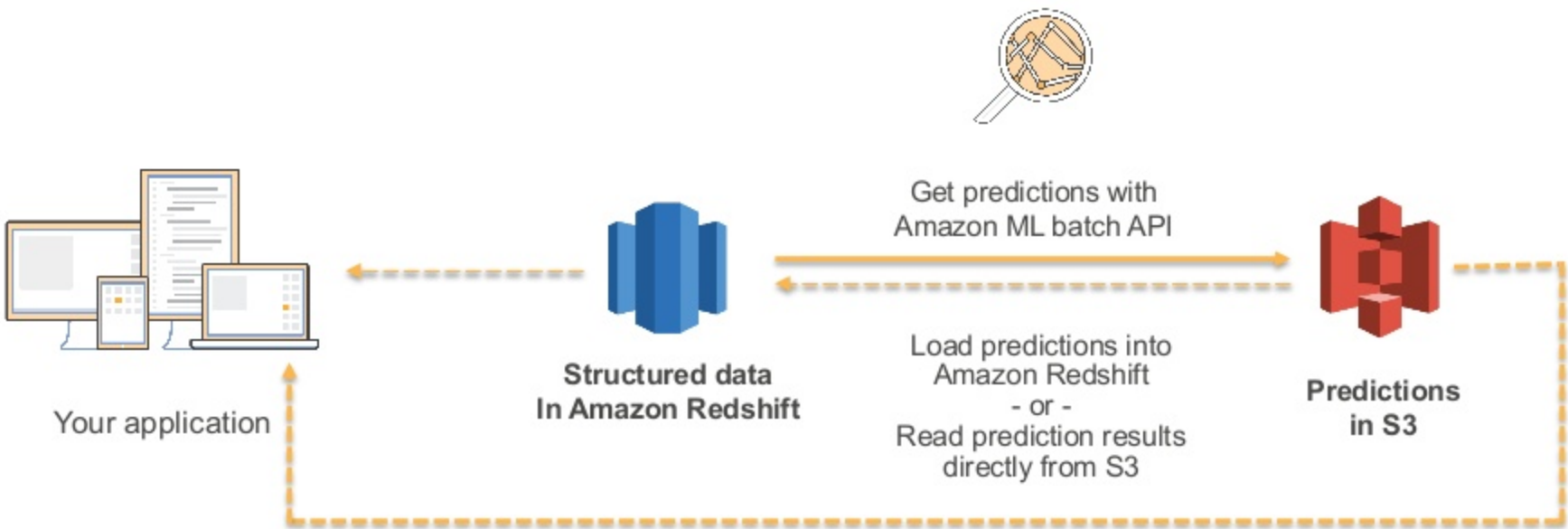
```
{
  'Prediction': {
    'predictedValue': 13.284348,
    'details': {
      'Algorithm': 'SGD',
      'PredictiveModelType': 'REGRESSION'
    }
  }
}
```

Architecture patterns for smart applications

Batch predictions with EMR



Batch predictions with Amazon Redshift



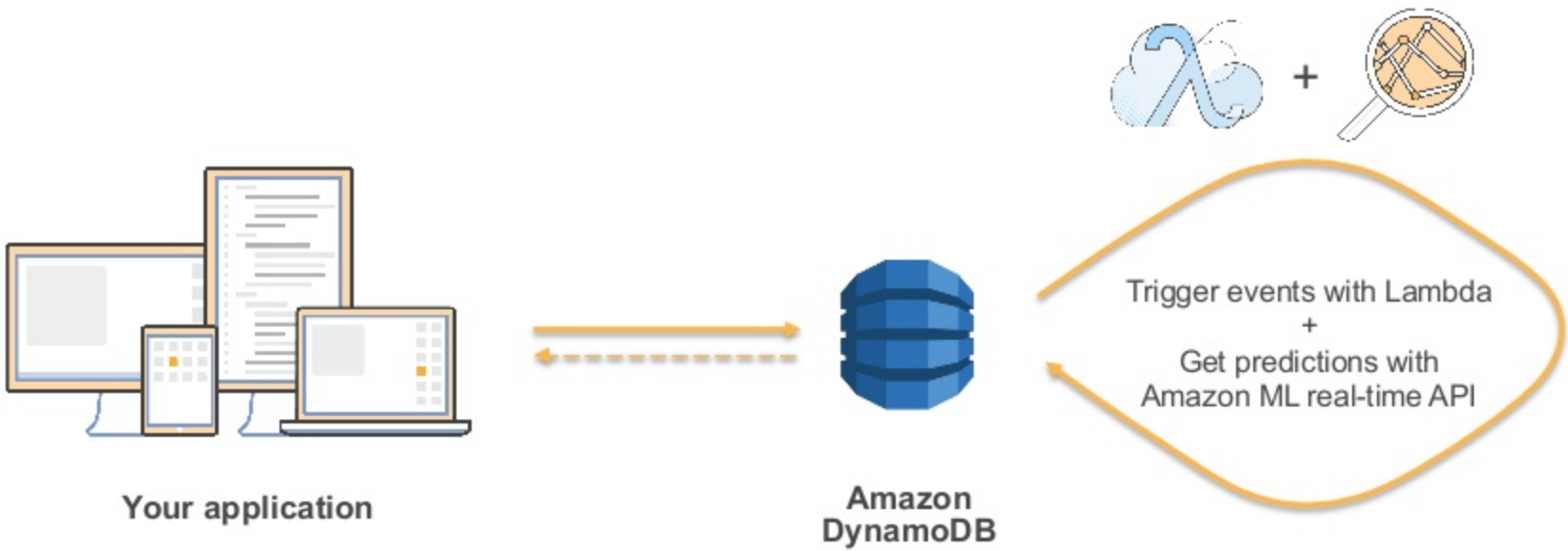
Real-time predictions for interactive applications

Get predictions with
Amazon ML real-time API

Your application

Amazon ML service

Adding predictions to an existing data flow

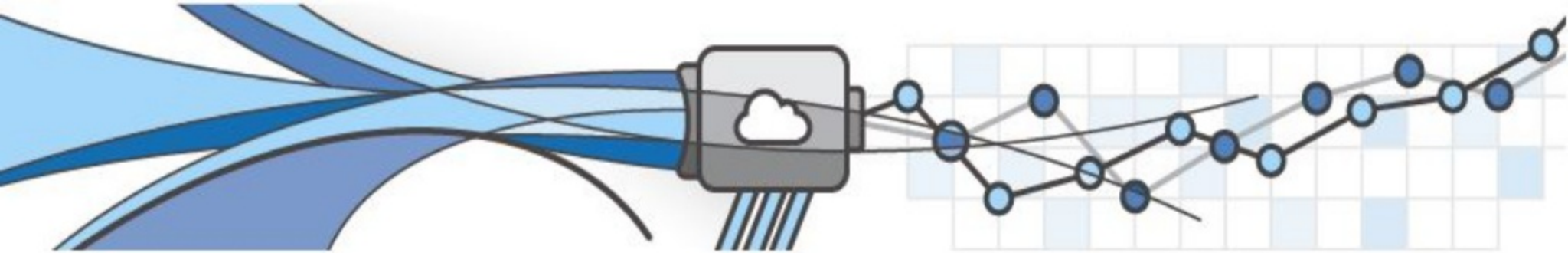


Q&A

http://aws.amazon.com/machine-learning

Next…

4:15 PM Building your first big data application on AWS



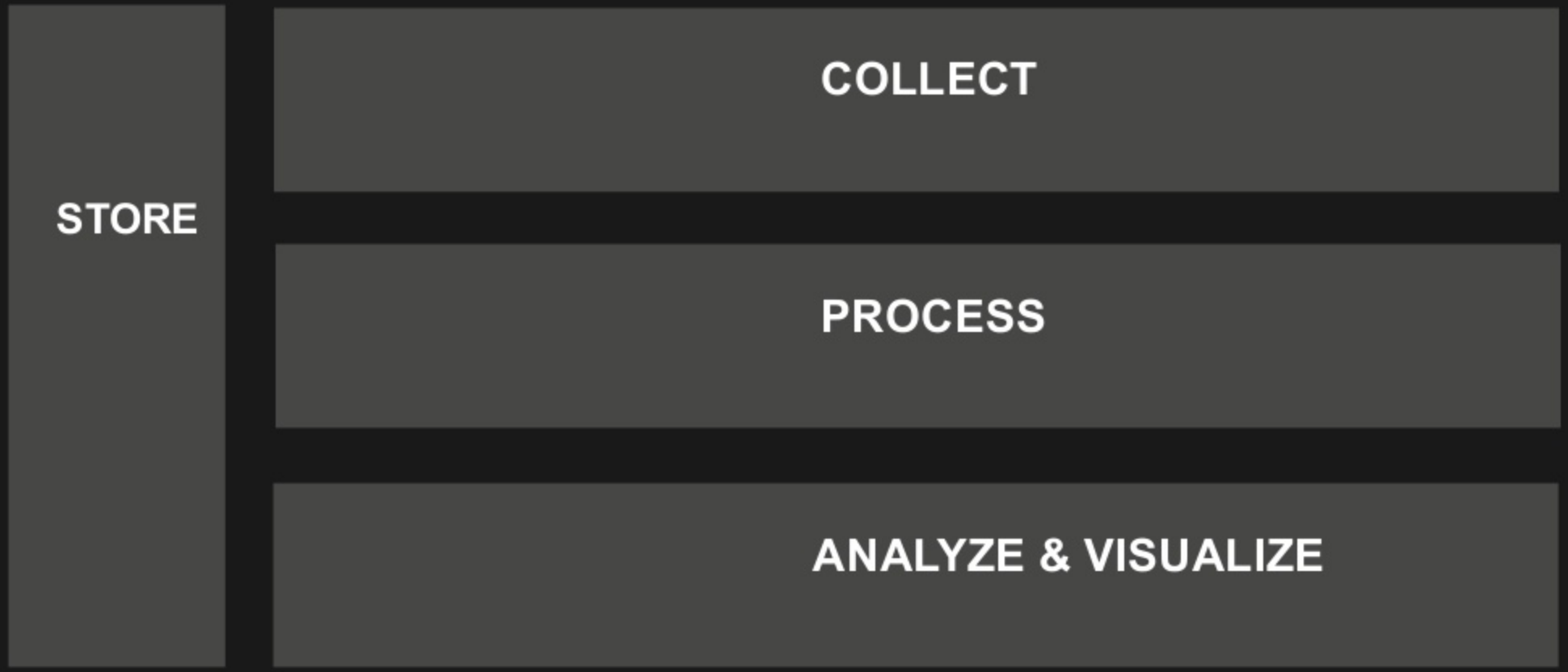
Building Your First Big Data Application on AWS

Radhika Ravirala

Solutions Architect

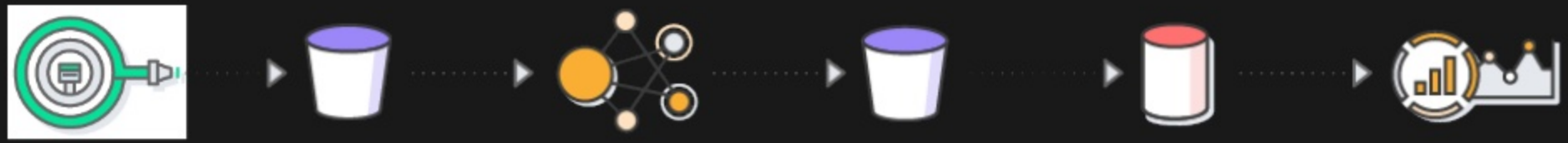
amazon web services

Your First Big Data Application on AWS



Your First Big Data Application on AWS

A Modern Take on the Classic Data Warehouse



Amazon Kinesis Firehose

Amazon S3

Amazon EMR

Amazon S3

Amazon Redshift

Amazon QuickSight

http://aws.amazon.com/big-data/use-cases/

Setting up the environment

Data Storage with Amazon S3

Download all the CLI steps: http://bit.ly/aws-big-data-steps

Create an Amazon S3 bucket to store the data collected with Amazon Kinesis Firehose

```
aws s3 mb s3://YOUR-S3-BUCKET-NAME
```

Access Control with IAM

Create an IAM role to allow Firehose to write to the S3 bucket

firehose-policy.json:

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"Service": "firehose.amazonaws.com"},
    "Action": "sts:AssumeRole"
  }
}
```

Access Control with IAM

Create an IAM role to allow Firehose to write to the S3 bucket

s3-rw-policy.json:

```
{ "Version": "2012-10-17",
  "Statement": {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::YOUR-S3-BUCKET-NAME",
        "arn:aws:s3:::YOUR-S3-BUCKET-NAME/*"
      ]
} }
```

Access Control with IAM

Create an IAM role to allow Firehose to write to the S3 bucket

```
aws iam create-role --role-name firehose-demo \
--assume-role-policy-document file://firehose-policy.json
```

Copy the value in “Arn” in the output,
e.g., *arn:aws:iam::123456789:role/firehose-demo*

```
aws iam put-role-policy --role-name firehose-demo \
--policy-name firehose-s3-rw \
--policy-document file://s3-rw-policy.json
```

Data Collection with Amazon Kinesis Firehose

Create a Firehose stream for incoming log data

```
aws firehose create-delivery-stream \
  --delivery-stream-name demo-firehose-stream \
  --s3-destination-configuration \
RoleARN=YOUR-FIREHOSE-ARN,\
BucketARN="arn:aws:s3:::YOUR-S3-BUCKET-NAME",\
Prefix=firehose\/,\
BufferingHints={IntervalInSeconds=60},\
CompressionFormat=GZIP
```

Data Processing with Amazon EMR

Launch an Amazon EMR cluster with Spark and Hive

```
aws emr create-cluster \
  --name "demo" \
  --release-label emr-4.5.0 \
  --instance-type m3.xlarge \
  --instance-count 2 \
  --ec2-attributes KeyName=YOUR-AWS-SSH-KEY \
  --use-default-roles \
  --applications Name=Hive Name=Spark Name=Zeppelin-Sandbox
```

Record your *ClusterId* from the output.

Data Analysis with Amazon Redshift

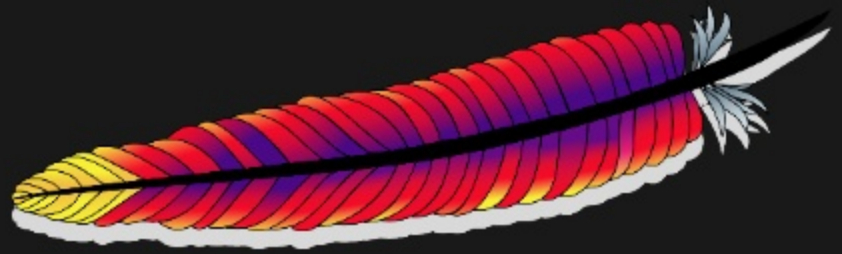
Create a single-node Amazon Redshift data warehouse:

```
aws redshift create-cluster \
  --cluster-identifier demo \
  --db-name demo \
  --node-type dc1.large \
  --cluster-type single-node \
  --master-username master \
  --master-user-password YOUR-REDSHIFT-PASSWORD \
  --publicly-accessible \
  --port 8192
```

Collect

Weblogs – Common Log Format (CLF)

```
75.35.230.210 - - [20/Jul/2009:22:22:42 -0700]
"GET /images/pigtrihawk.jpg HTTP/1.1" 200 29236 "http://
www.swivel.com/graphs/show/1163466"
"Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.11)
Gecko/2009060215 Firefox/3.0.11 (.NET CLR 3.5.30729)"
```



Writing into Amazon Kinesis Firehose

Download the demo weblog: `http://bit.ly/aws-big-data`

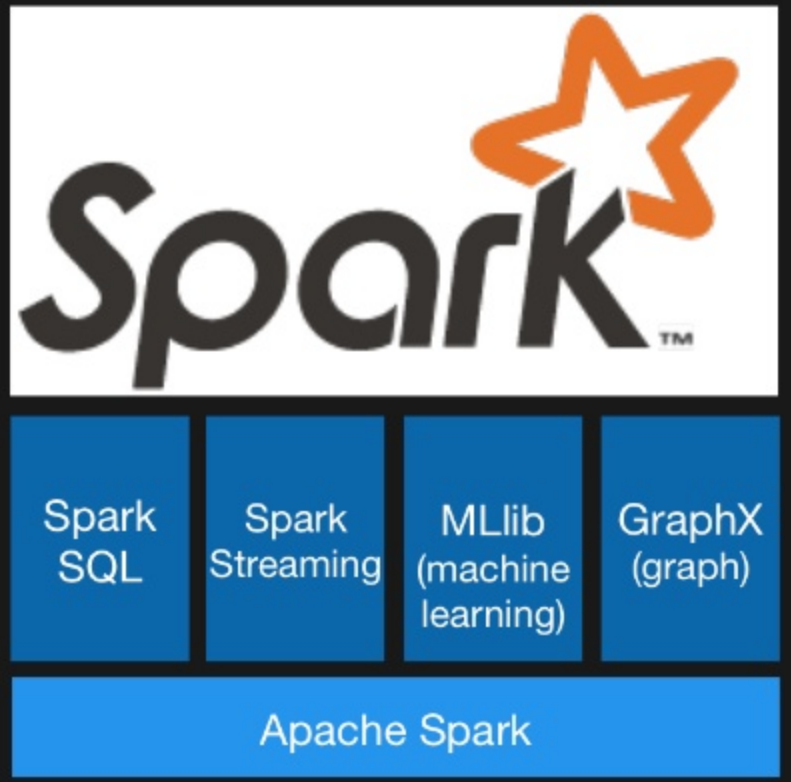
Open Python and run the following code to import the log into the stream:

```
import boto3
iam = boto3.client('iam')
firehose = boto3.client('firehose')

with open('weblog', 'r') as f:
    for line in f:
        firehose.put_record(
                DeliveryStreamName='demo-firehose-stream',
                Record={'Data': line}
                )
        print 'Record added'
```

Process

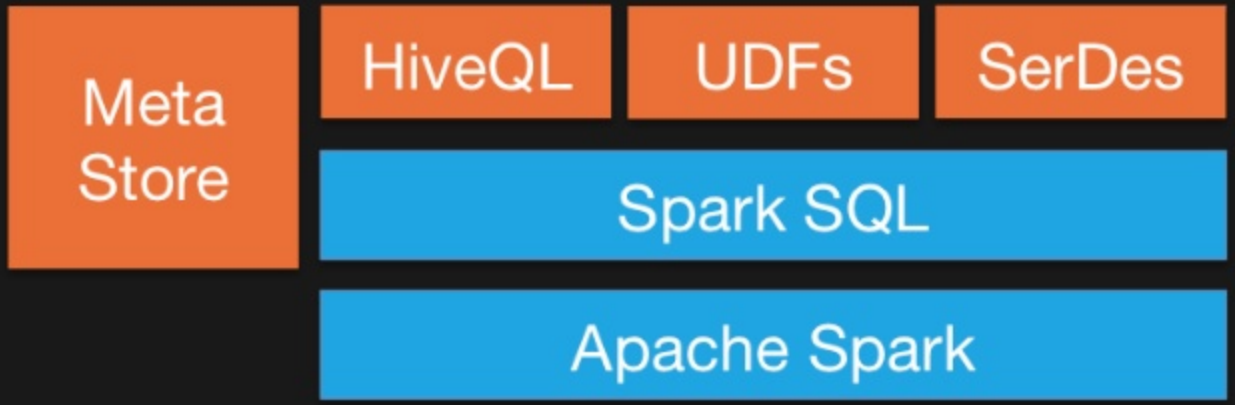
Apache Spark

- Fast, general purpose engine for large-scale data processing
- Write applications quickly in Java, Scala, or Python
- Combine SQL, streaming, and complex analytics



Spark SQL

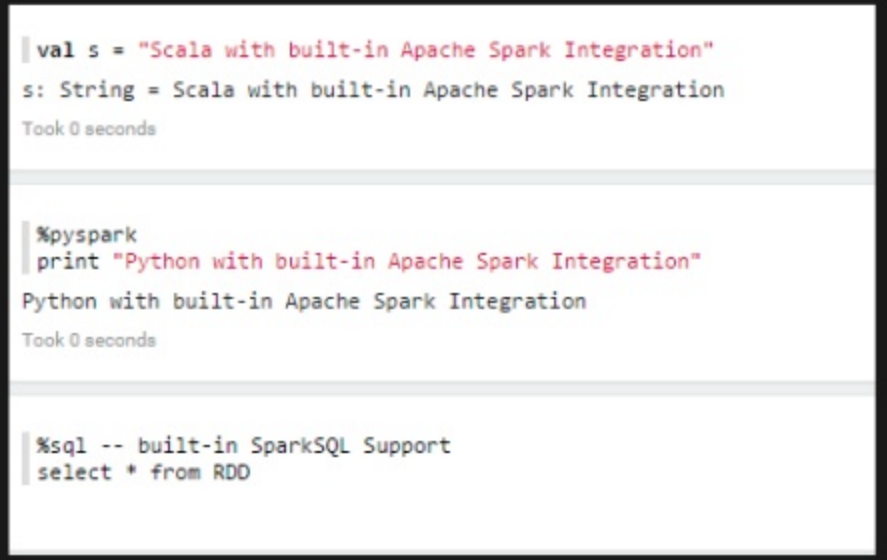
Spark's module for working with structured data using SQL



Run unmodified Hive queries on existing data

Apache Zeppelin

- Web-based notebook for interactive analytics
- Multiple language backend
- Apache Spark integration
- Data visualization
- Collaboration



https://zeppelin.incubator.apache.org/

View the Output Files in Amazon S3

After about 1 minute, you should see files in your S3 bucket:

```
aws s3 ls s3://YOUR-S3-BUCKET-NAME/firehose/ --recursive
```

Connect to Your EMR Cluster and Zeppelin

```
aws emr describe-cluster --cluster-id YOUR-EMR-CLUSTER-ID
```

Copy the *MasterPublicDnsName*. Use port forwarding so you can access Zeppelin at http://localhost:18890 on your local machine.

```
ssh -i PATH-TO-YOUR-SSH-KEY -L 18890:localhost:8890 \
hadoop@YOUR-EMR-DNS-NAME
```

Open Zeppelin with your local web browser and create a new “Note”:
`http://localhost:18890`

Exploring the Data in Amazon S3 using Spark

Download the Zeppelin notebook: http://bit.ly/aws-big-data-zeppelin

```
// Load all the files from S3 into a RDD
val accessLogLines = sc.textFile("s3://YOUR-S3-BUCKET-NAME/firehose/*/*/*/*")

// Count the lines
accessLogLines.count

// Print one line as a string
accessLogLines.first

// delimited by space so split them into fields
var accessLogFields = accessLogLines.map(_.split(" ").map(_.trim))

// Print the fields of a line
accessLogFields.first
```

Combine Fields: “A, B, C” → “A B C”

```
var accessLogColumns = accessLogFields
    .map( arrayOfFields => { var temp1 =""; for (field <- arrayOfFields) yield {
        var temp2 = ""
        if (temp1.replaceAll("\\[","\"").startsWith("\"") && !temp1.endsWith("\""))
            temp1 = temp1 + " " + field.replaceAll("\\[|\\]","\"")
        else temp1 = field.replaceAll("\\[|\\]","\"")
        temp2 = temp1
        if (temp1.endsWith("\"")) temp1 = ""
        temp2
    }})
    .map( fields => fields.filter(field => (field.startsWith("\"") &&
field.endsWith("\"")) || !field.startsWith("\"") ))
    .map(fields => fields.map(_.replaceAll("\"","")))
```

Create a Data Frame and Transform the Data

```
import java.sql.Timestamp
import java.net.URL
case class accessLogs(
  ipAddress: String,
  requestTime: Timestamp,
  requestMethod: String,
  requestPath: String,
  requestProtocol: String,
  responseCode: String,
  responseSize: String,
  referrerHost: String,
  userAgent: String
)
```

Create a Data Frame and Transform the Data

```
val accessLogsDF = accessLogColumns.map(line => {
    var ipAddress       = line(0)
    var requestTime     = new Timestamp(new java.text.SimpleDateFormat("dd/MMM/
yyyy:HH:mm:ss Z").parse(line(3)).getTime())
    var requestString   = line(4).split(" ").map(_.trim())
    var requestMethod   = if (line(4).toString() != "-") requestString(0) else ""
    var requestPath     = if (line(4).toString() != "-") requestString(1) else ""
    var requestProtocol = if (line(4).toString() != "-") requestString(2) else ""
    var responseCode    = line(5).replaceAll("-","")
    var responseSize    = line(6).replaceAll("-","")
    var referrerHost    = line(7)
    var userAgent       = line(8)
    accessLogs(ipAddress, requestTime, requestMethod, requestPath,
requestProtocol,responseCode, responseSize, referrerHost, userAgent)
}).toDF()
```

Create an External Table Backed by Amazon S3

```
%sql
CREATE EXTERNAL TABLE access_logs
(
 ip_address String,
 request_time Timestamp,
 request_method String,
 request_path String,
 request_protocol String,
 response_code String,
 response_size String,
 referrer_host String,
 user_agent String
)
PARTITIONED BY (year STRING,month STRING, day STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE
LOCATION 's3://YOUR-S3-BUCKET-NAME/access-log-processed'
```

Configure Hive Partitioning and Compression

```
// set up Hive's "dynamic partitioning"
%sql
SET hive.exec.dynamic.partition=true

// compress output files on Amazon S3 using Gzip
%sql
SET hive.exec.compress.output=true

%sql
SET mapred.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec

%sql
SET io.compression.codecs=org.apache.hadoop.io.compress.GzipCodec

%sql
SET hive.exec.dynamic.partition.mode=nonstrict;
```

Write Output to Amazon S3

```
import org.apache.spark.sql.SaveMode
accessLogsDF
    .withColumn("year", year(accessLogsDF("requestTime")))
    .withColumn("month", month(accessLogsDF("requestTime")))
    .withColumn("day", dayofmonth(accessLogsDF("requestTime")))
    .write
    .partitionBy("year","month","day")
    .mode(SaveMode.Overwrite)
    .insertInto("access_logs")
```

Query the Data Using Spark SQL

```
// Check the count of records
%sql
select count(*) from access_log_processed


// Fetch the first 10 records
%sql
select * from access_log_processed limit 10
```

View the Output Files in Amazon S3

Leave Zeppelin and go back to the console…

List the partition prefixes and output files:

```
aws s3 ls s3://YOUR-S3-BUCKET-NAME/access-log-processed/ \
  --recursive
```

Analyze

Connect to Amazon Redshift

Using the PostgreSQL CLI

```
psql -h YOUR-REDSHIFT-ENDPOINT \
     -p 8192 -U master demo
```

Or use any JDBC or ODBC SQL client with the PostgreSQL 8.x drivers or native Amazon Redshift support

- *Aginity Workbench for Amazon Redshift*
- *SQL Workbench/J*

Create an Amazon Redshift Table to Hold Your Data

```
CREATE TABLE accesslogs
(
        host_address varchar(512),
        request_time timestamp,
        request_method varchar(5),
        request_path varchar(1024),
        request_protocol varchar(10),
        response_code Int,
        response_size Int,
        referrer_host varchar(1024),
        user_agent varchar(512)
)
DISTKEY(host_address)
SORTKEY(request_time);
```

Loading Data into Amazon Redshift

"COPY" command loads files in parallel from Amazon S3:

```
COPY accesslogs
FROM 's3://YOUR-S3-BUCKET-NAME/access-log-processed'
CREDENTIALS
'aws_iam_role=arn:aws:iam::YOUR-AWS-ACCOUNT-ID:role/ROLE-NAME'
DELIMITER '\t'
MAXERROR 0
GZIP;
```

Amazon Redshift Test Queries

```
-- find distribution of response codes over days
SELECT TRUNC(request_time), response_code, COUNT(1) FROM
accesslogs GROUP BY 1,2 ORDER BY 1,3 DESC;

-- find the 404 status codes
SELECT COUNT(1) FROM accessLogs WHERE response_code = 404;

-- show all requests for status as PAGE NOT FOUND
SELECT TOP 1 request_path,COUNT(1) FROM accesslogs WHERE
response_code = 404 GROUP BY 1 ORDER BY 2 DESC;
```

Visualize the Results

DEMO
Amazon QuickSight

Automating the Big Data Application

Learn from big data experts

blogs.aws.amazon.com/bigdata

Thank you to our Sponsor