AWS re:Invent

BDM206

Understanding IoT Data

How to Leverage Amazon Kinesis in Building an IoT Analytics Platform on AWS

Daniel Zoltak, Solutions Architect, AWS
Marc Teichtahl, Solutions Architect, AWS
Tim Bart, CTO, Hello

November 29, 2016

amazon web services

What to expect from the session

Together, we will:

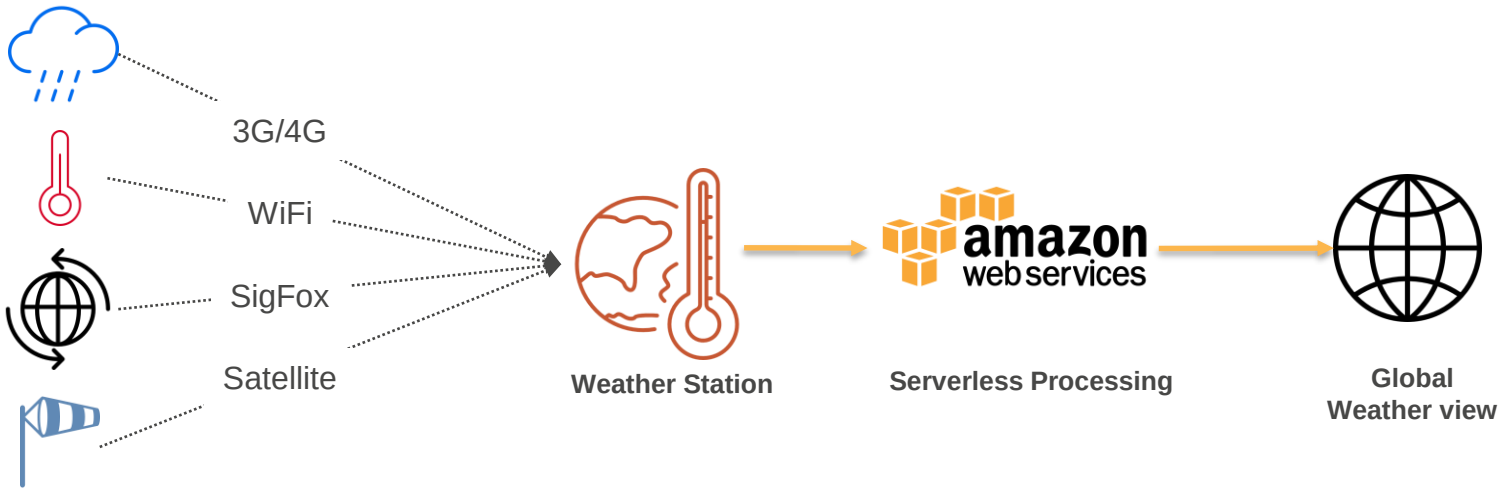
- Explore two real use cases of **IoT Analytics** using the **Amazon Kinesis** family of services.
- See a demo of IoT and Amazon Kinesis in action.
- Take a deep dive into underlying **reference architectures** and **implementation**.
- Hear from an AWS customer **hello, an IoT company**, about their use case, journey, and implementation.

What to expect from the session

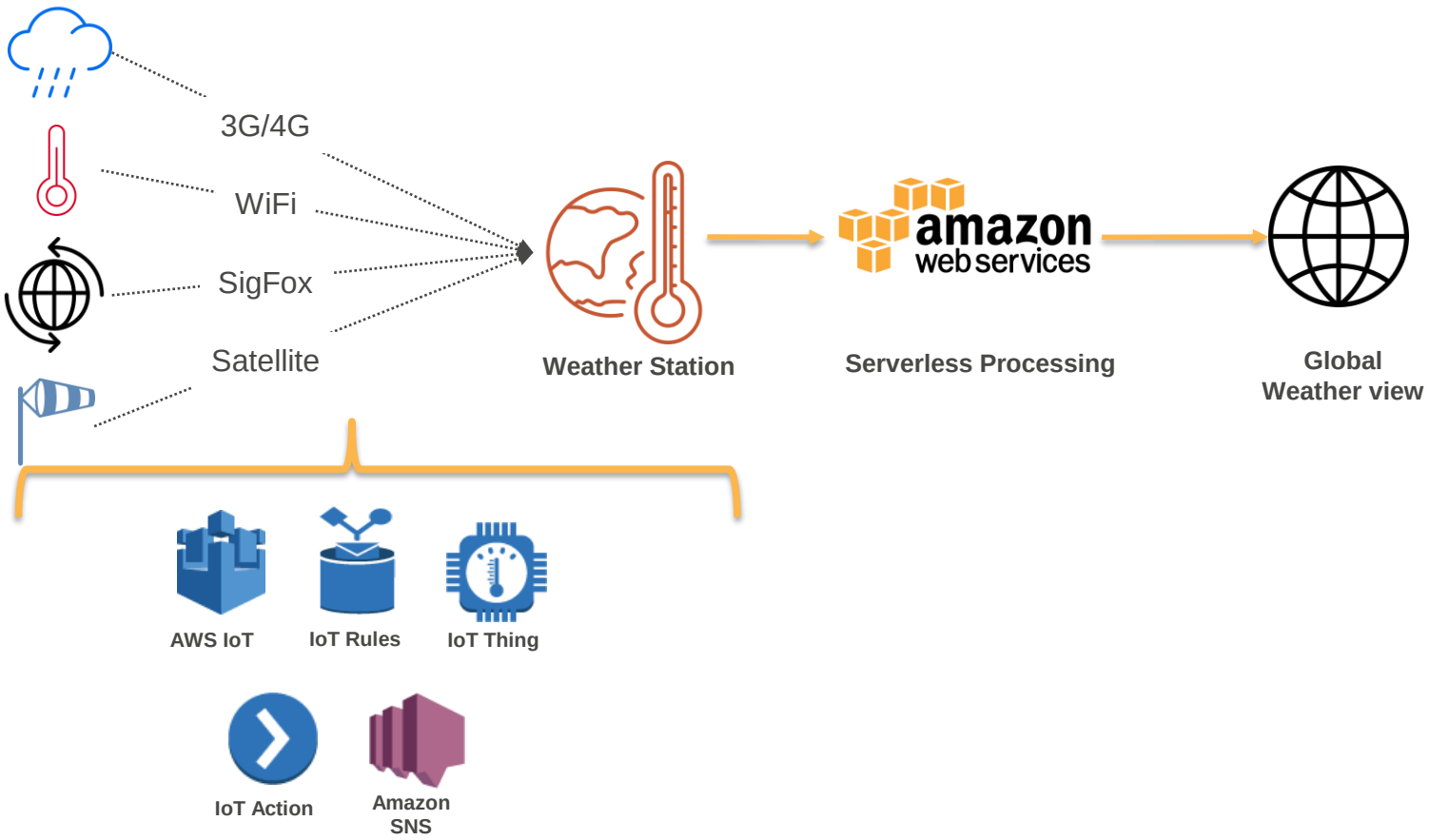
By the end of this session, you will:

- Have an appreciation of the AWS services required to build a **serverless** IoT **analytics** platform.
- Be able to describe the role and functionality of Amazon Kinesis **Firehose**, Amazon Kinesis **Streams**, and Amazon Kinesis **Analytics**.
- Understand how to **acquire**, **process**, and **store** IoT data.

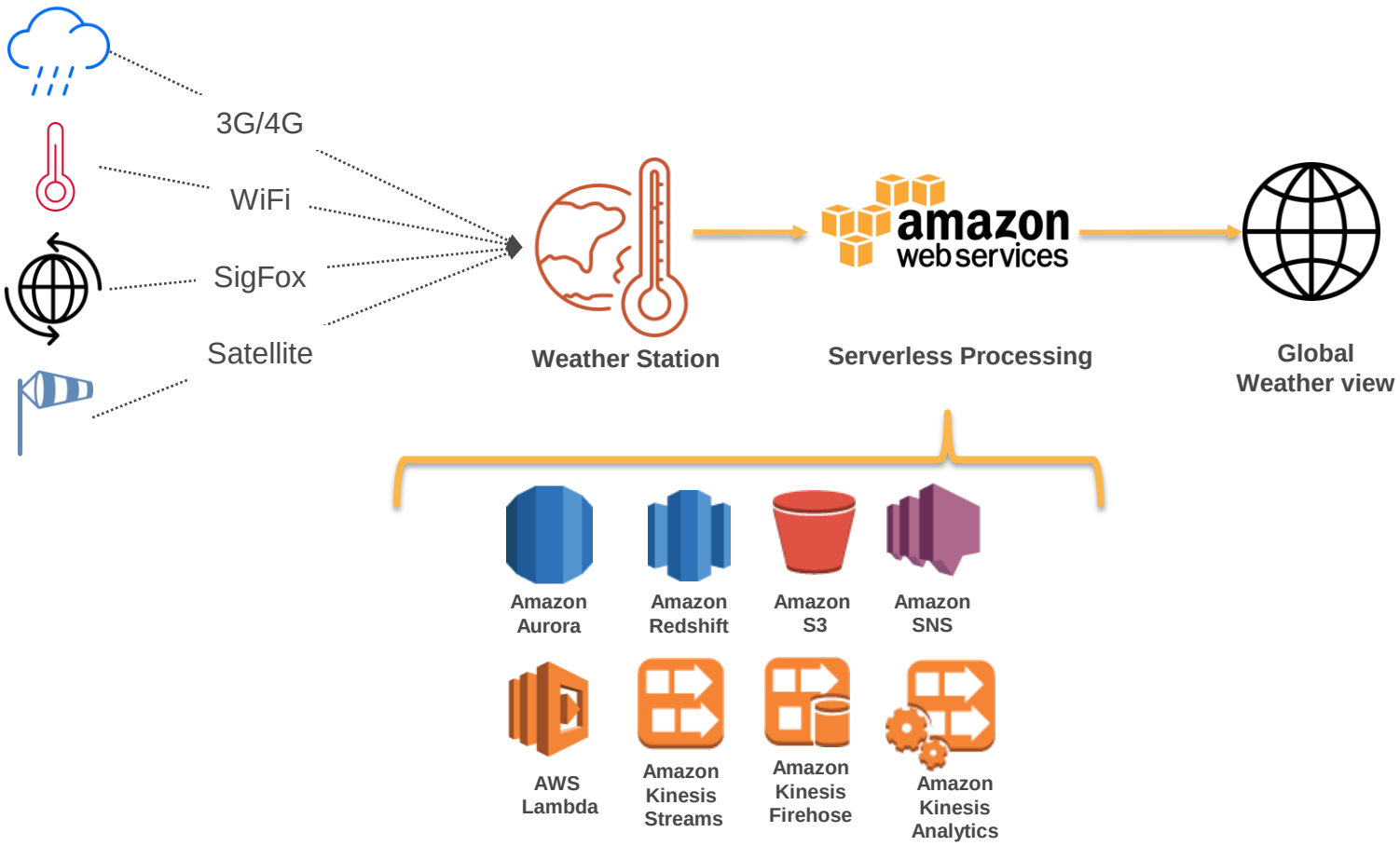
What you are about to see



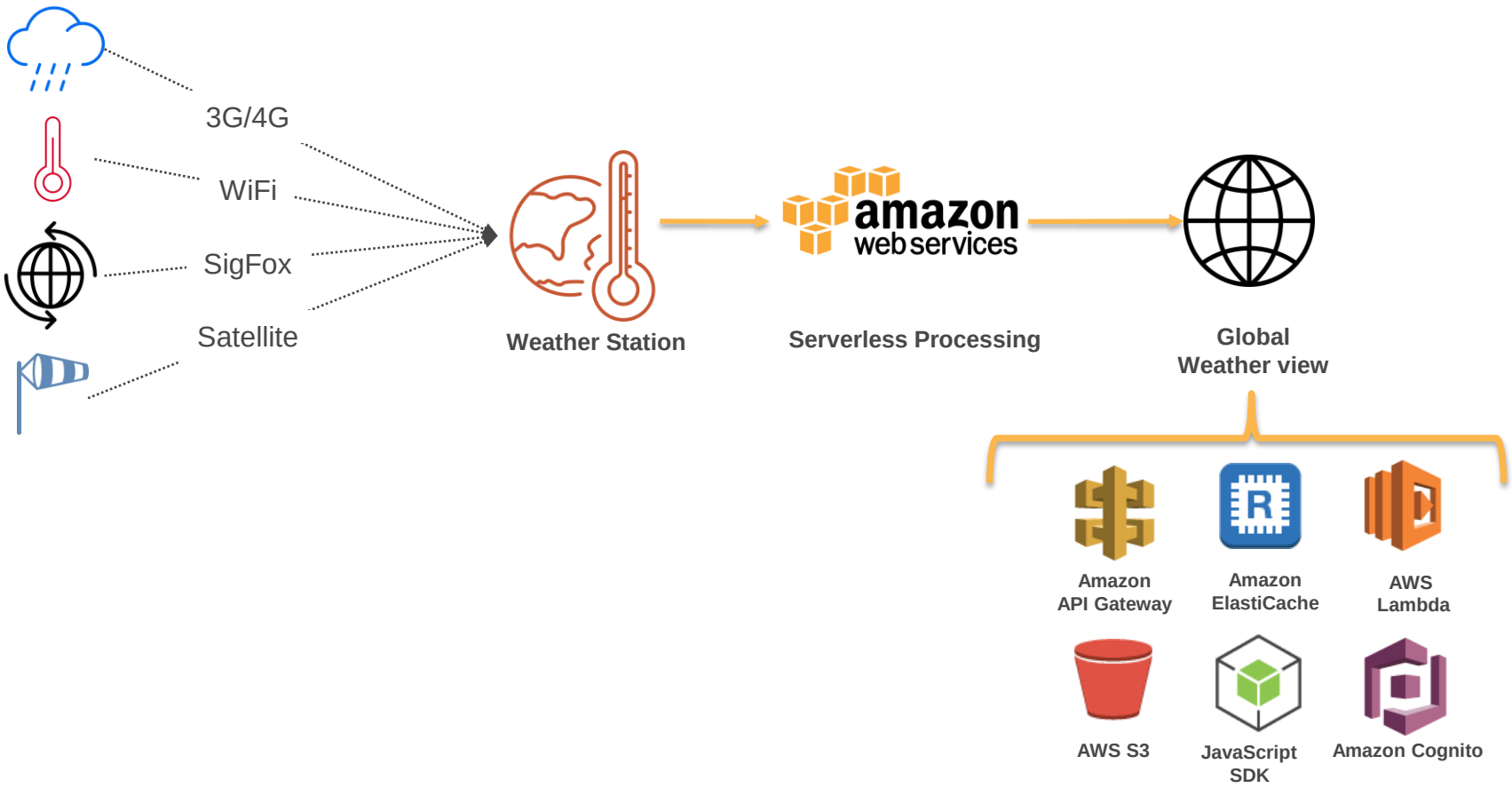
What you are about to see



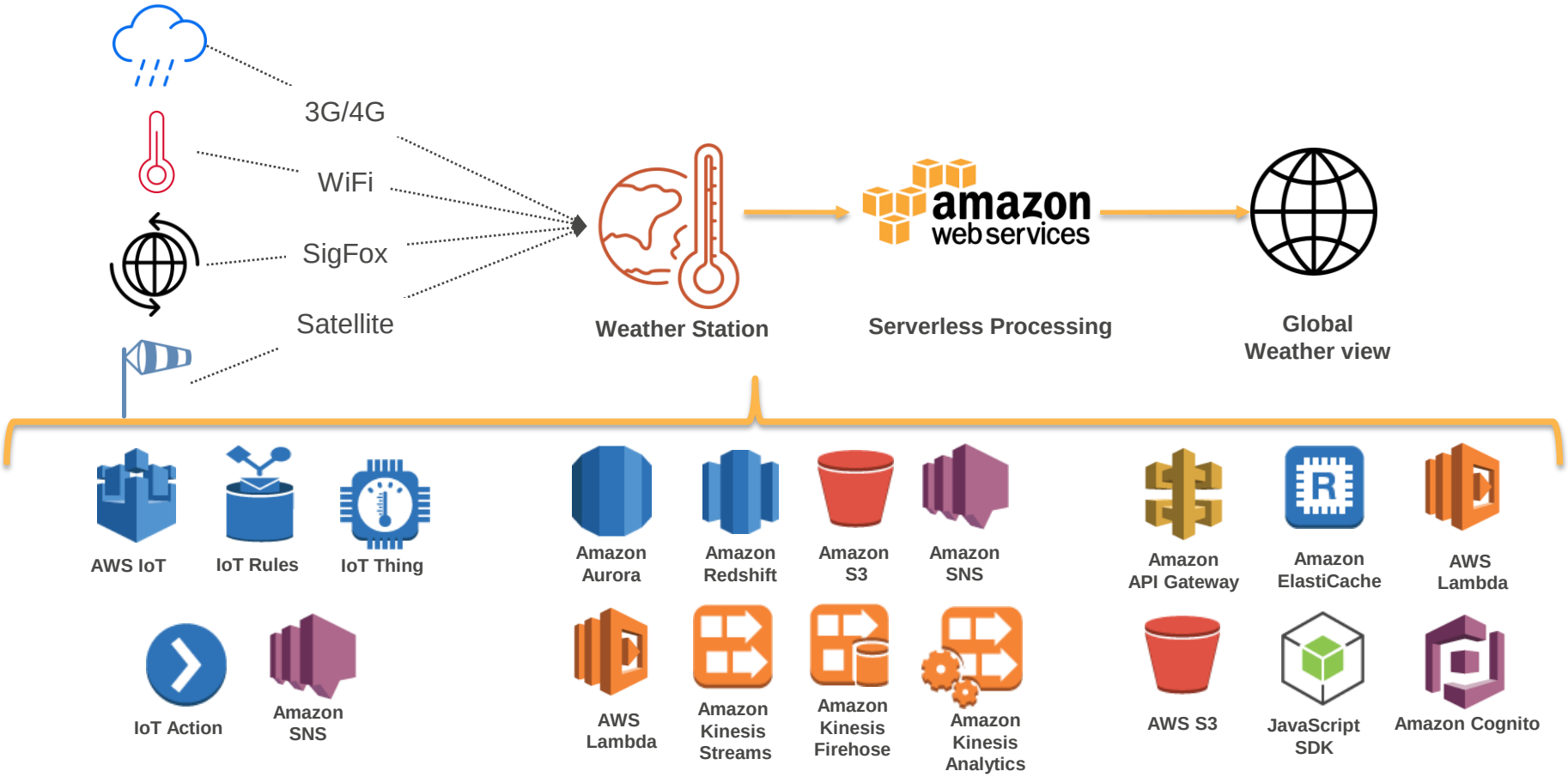
What you are about to see



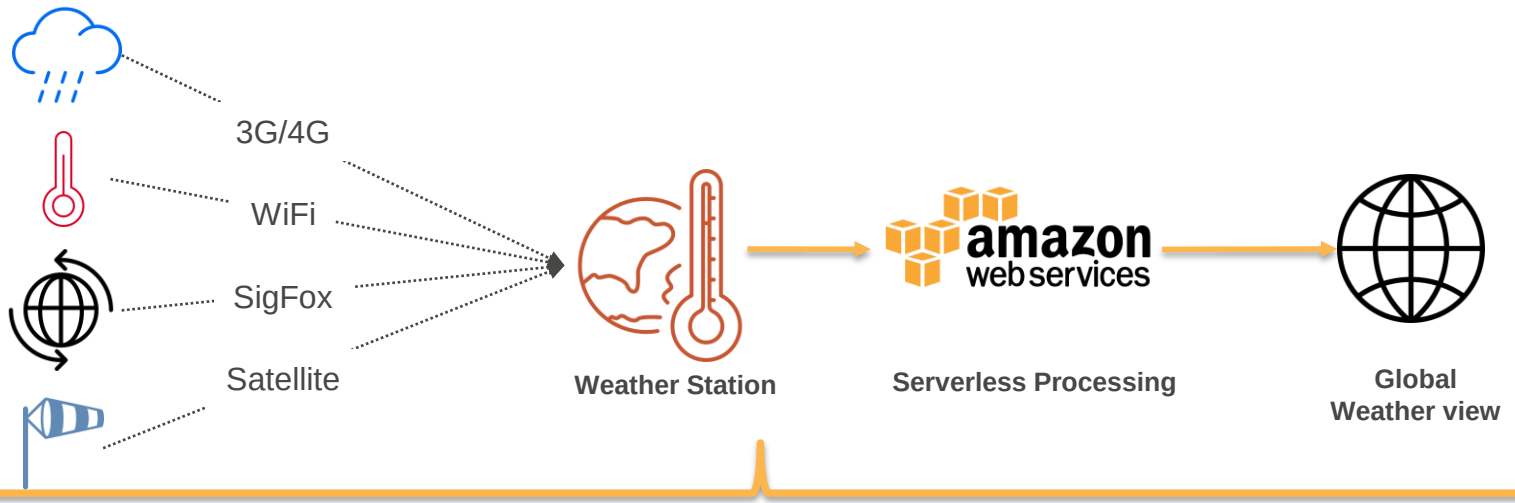
What you are about to see



What you are about to see



What you are about to see



10 AWS features and services

&

0 servers to manage

Let’s see this in action

DEMO

What do our customers ask for?

- Our customers ask us to help them
  - Ingest **large volumes** of real-time data from a large fleet of distributed IoT devices at scale.
  - Perform advanced **analytics** of streaming data in real-time.
  - **Process** and store large volumes of data.
  - Eliminate capacity planning, **scaling**, and the management of infrastructure.

Why did our customers ask?

Designing for failure in global, real-time, distributed systems is hard.

Why did our customers ask?

Designing for failure in global, real-time, distributed systems is hard.

Infrastructure required to process billions of devices sending trillions of messages is expensive.

Why did our customers ask?

Designing for failure in global, real-time, distributed systems is hard.

Infrastructure required to process billions of devices sending trillions of messages is expensive.

Management overhead and scale limitations impede innovation.

Why did our customers ask?

Let AWS do the undifferentiated heavy lifting for you

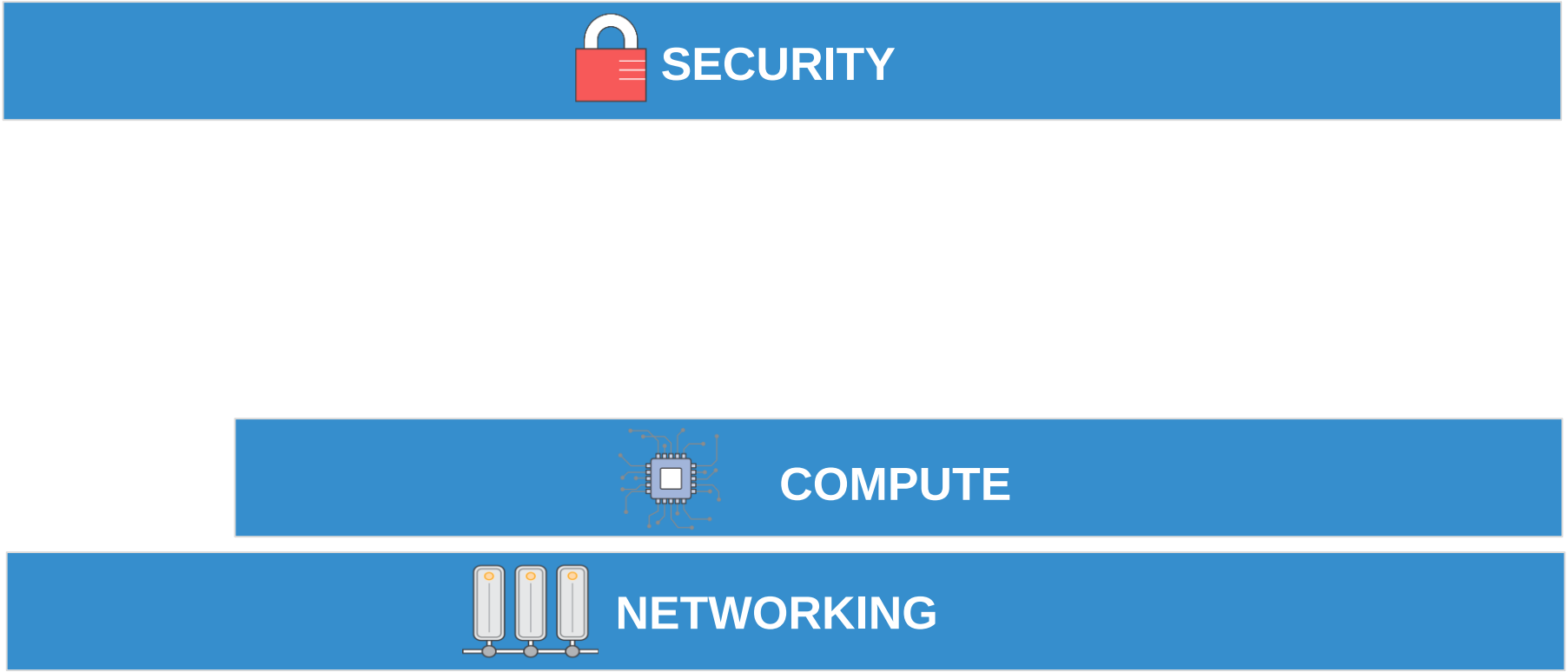
Reference Model

Reference Model

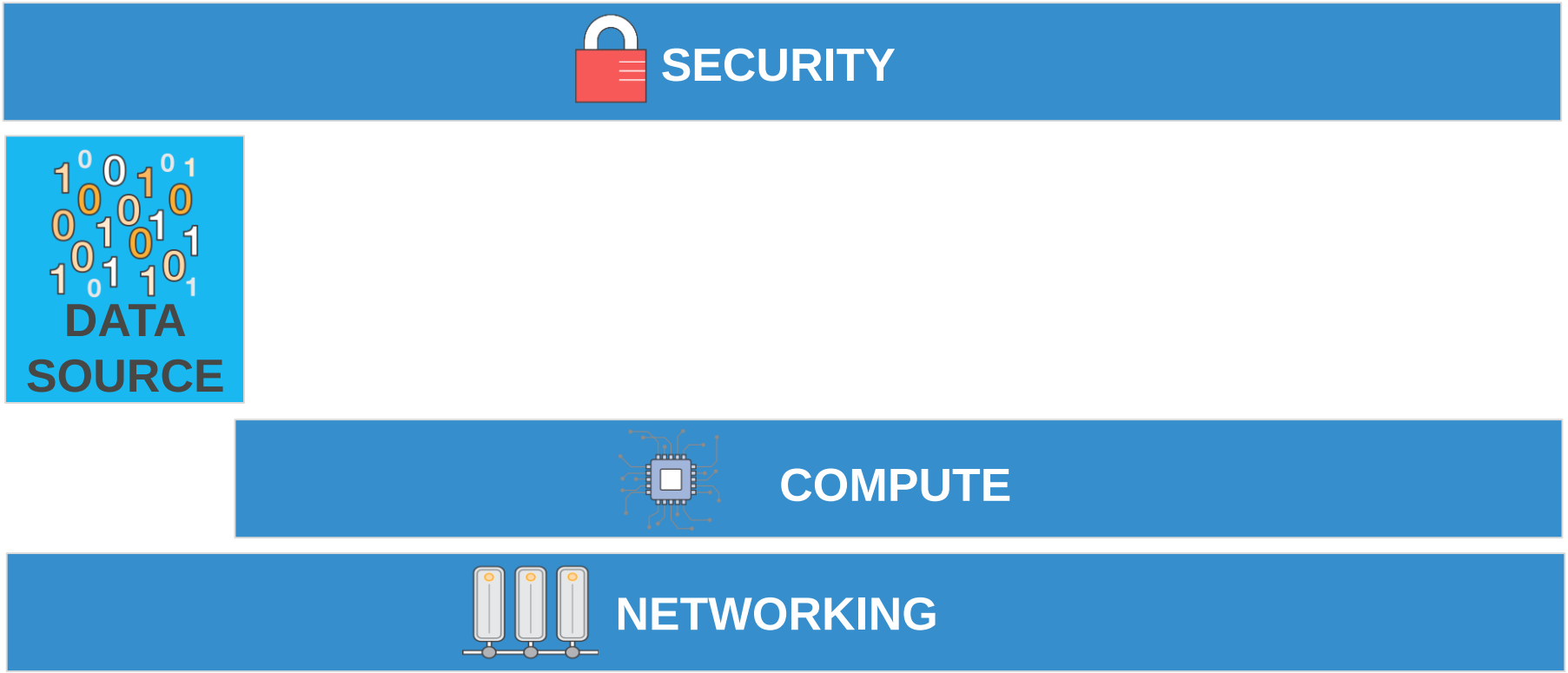
SECURITY

NETWORKING

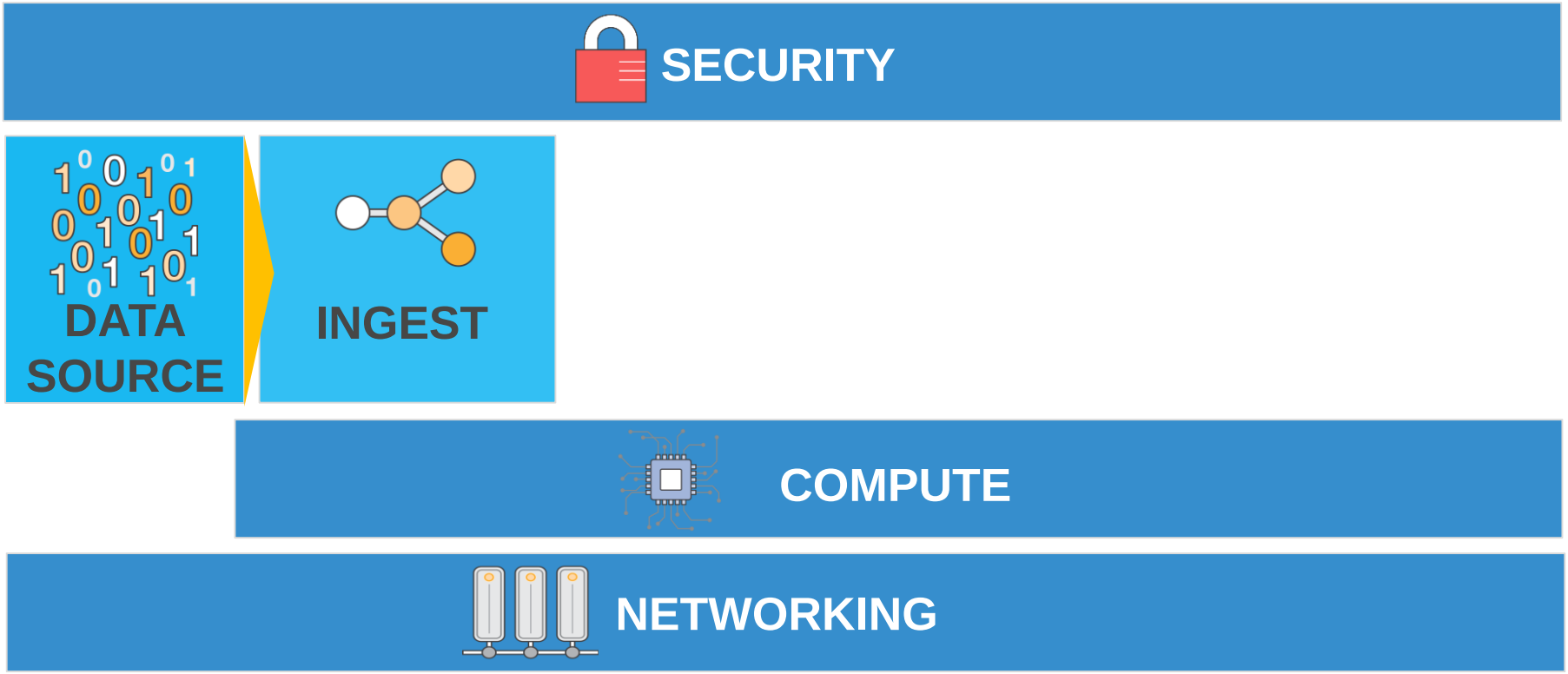
Reference Model



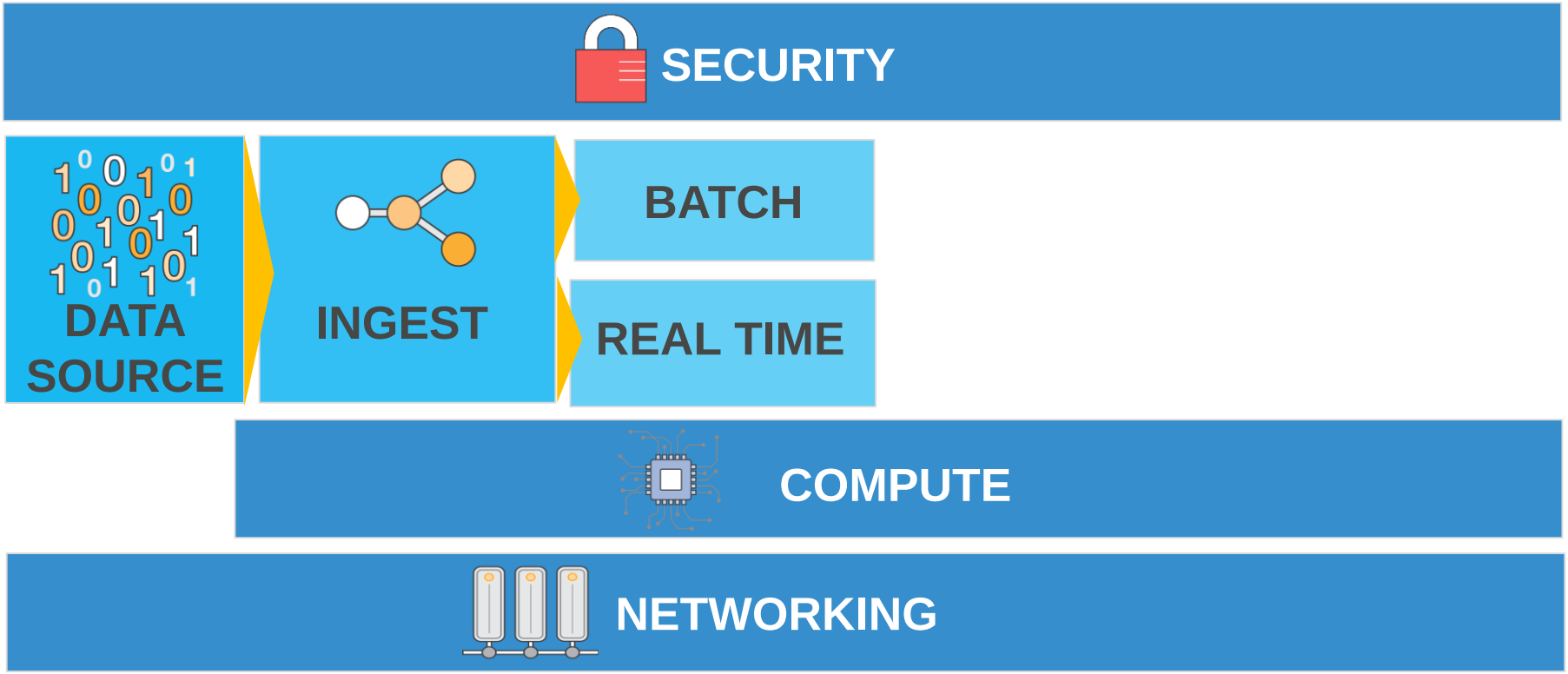
Reference Model



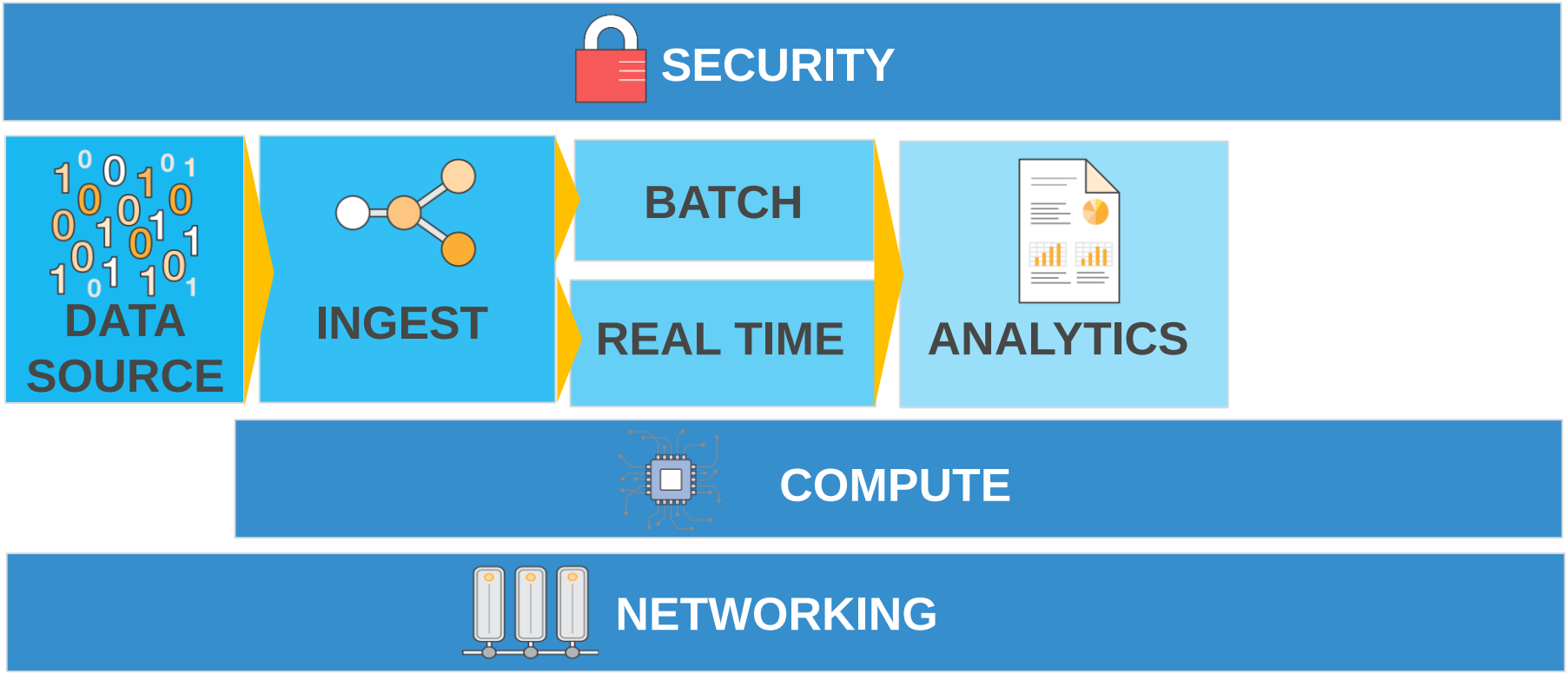
Reference Model



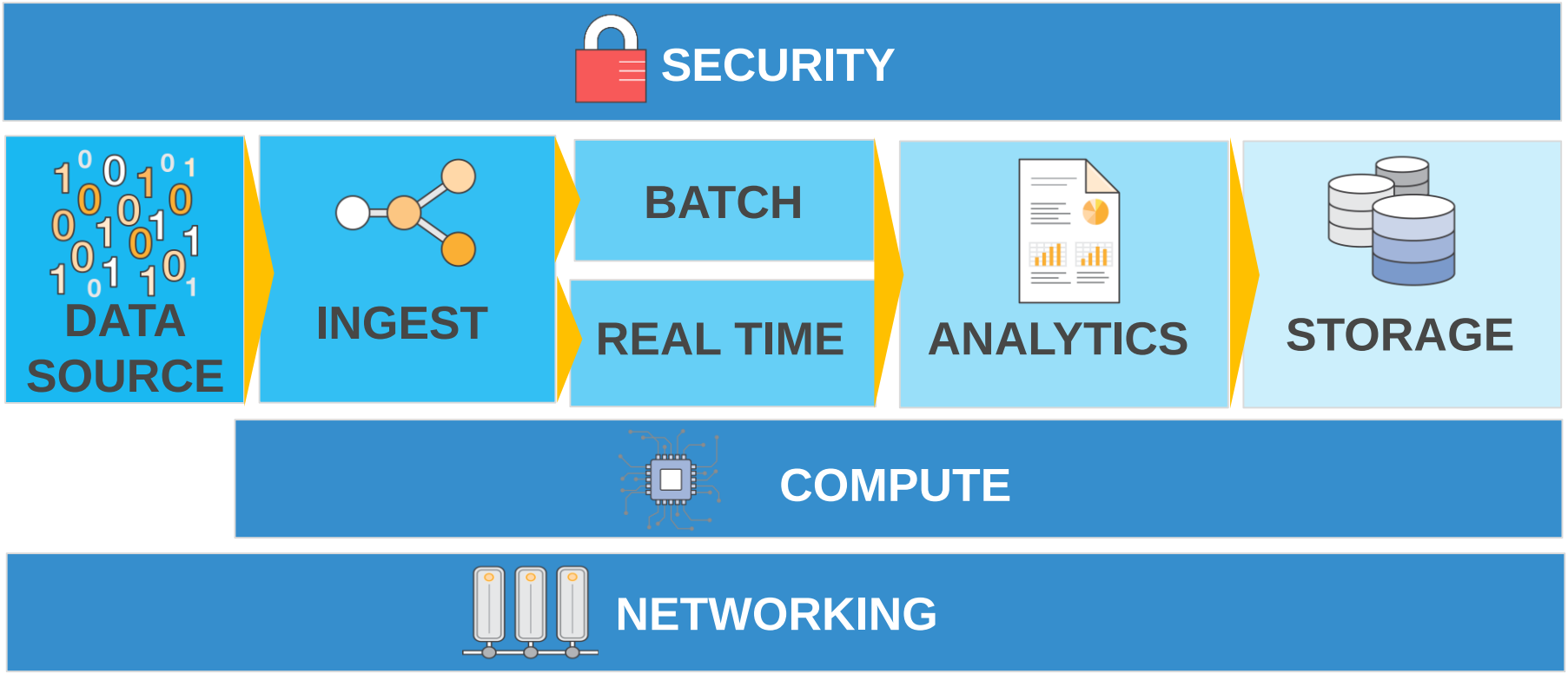
Reference Model



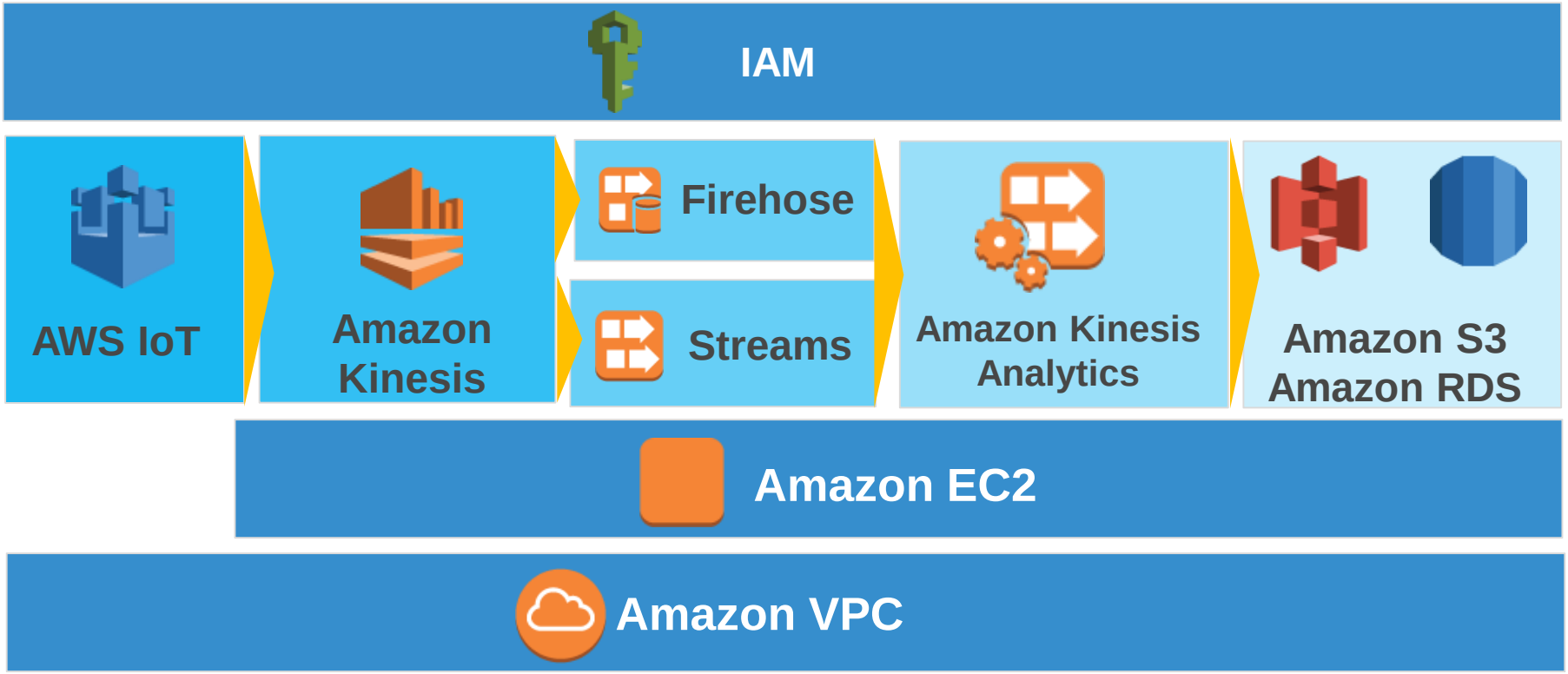
Reference Model



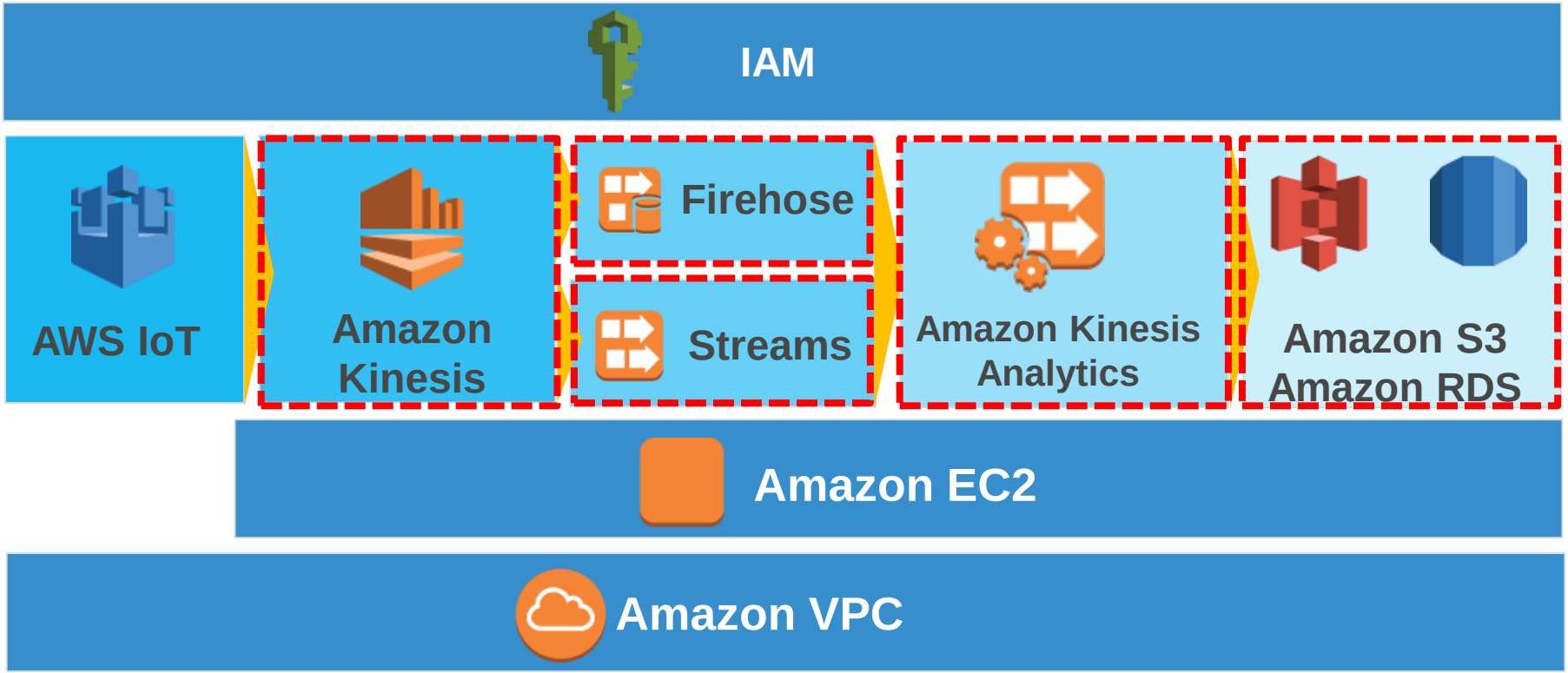
Reference Model



Reference Model

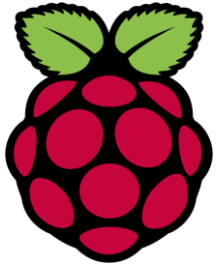


Reference Model - Focus Today

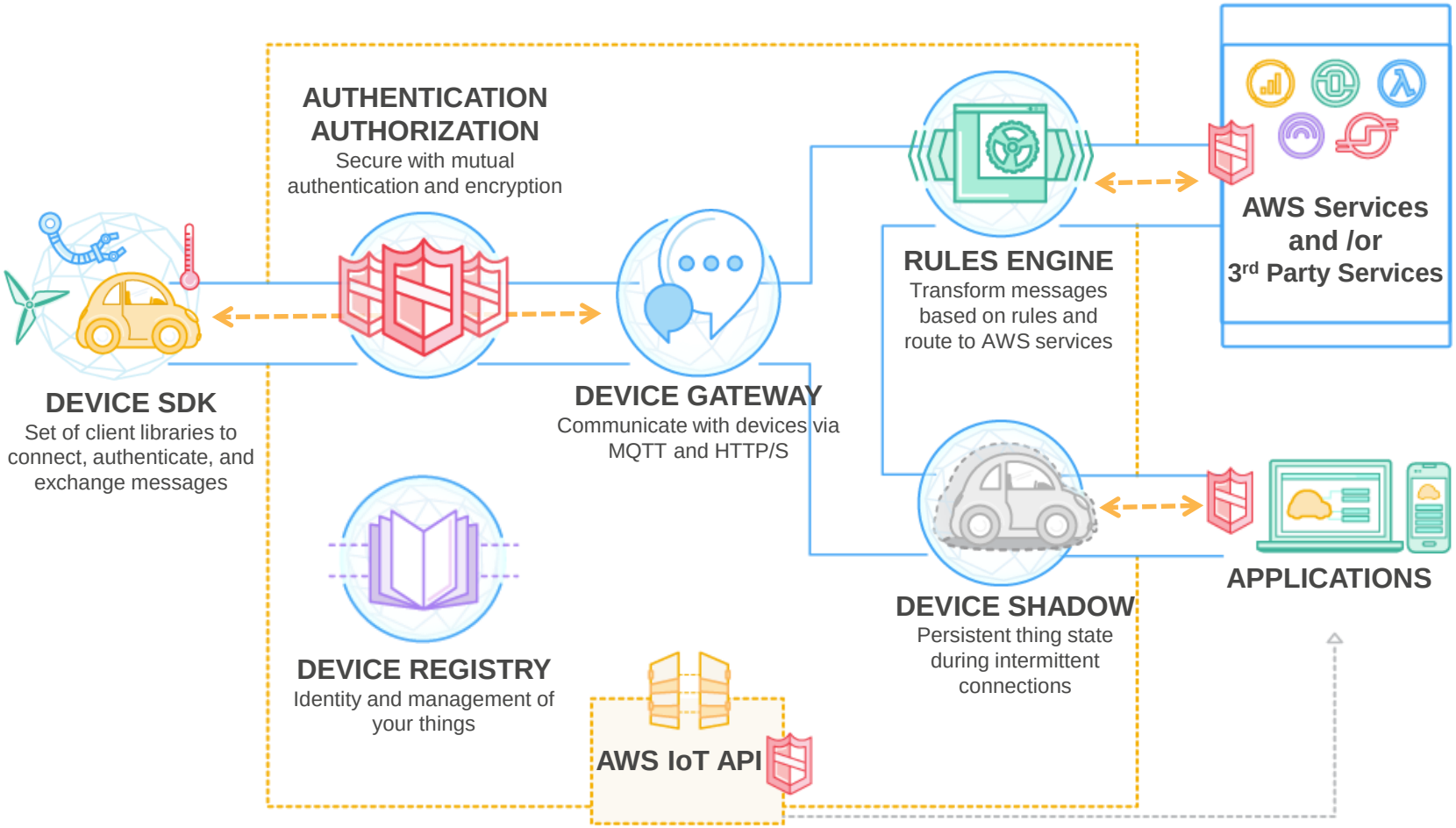


What Is An IoT “Thing”?

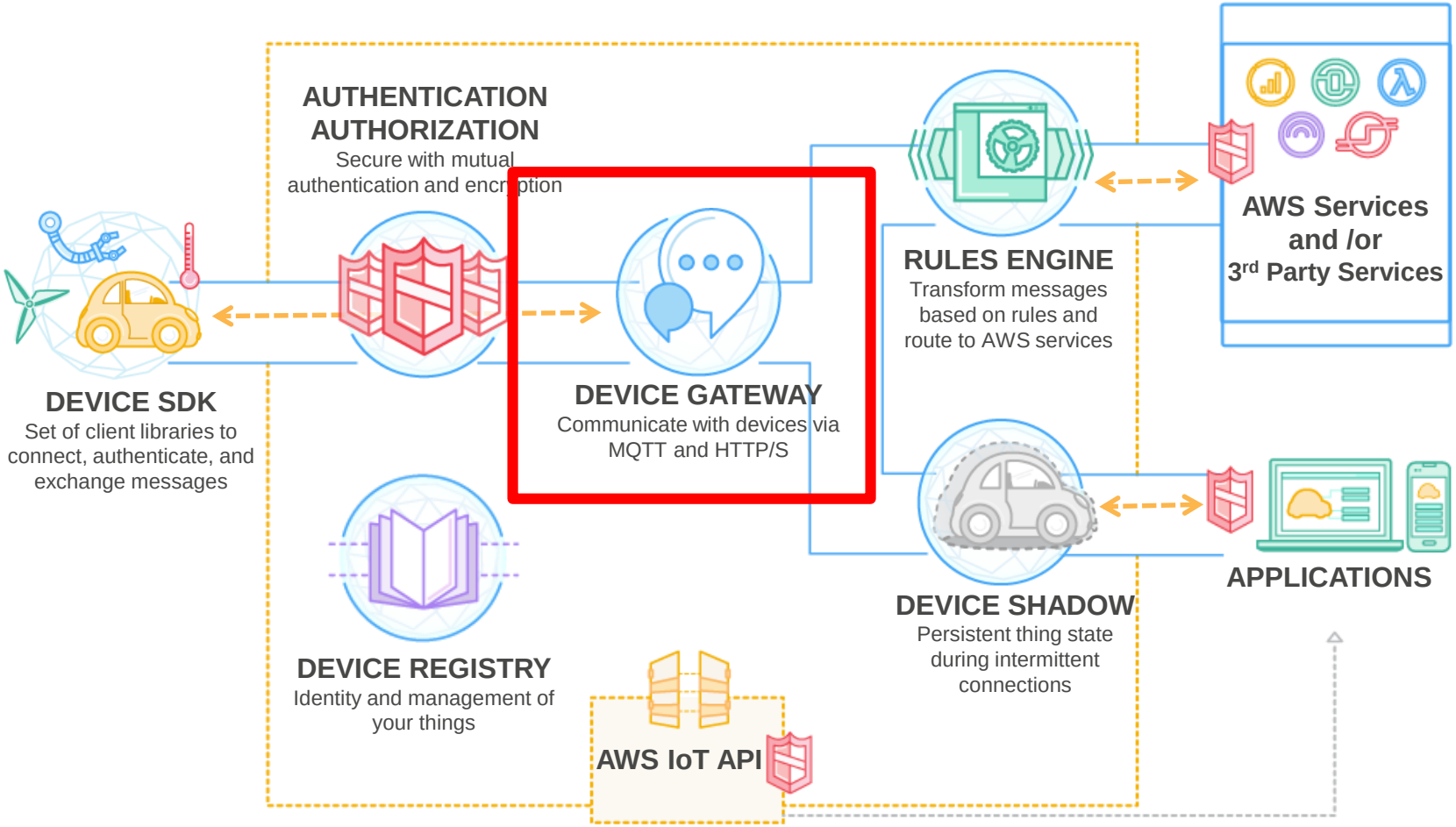
- Mobile Devices
  - IOS, Android, Kindle, Tablets.
- Maker Devices
  - Arduino, Raspberry Pi, Intel Edison.
- Embedded devices and wearables
  - Health and fitness management; safety and tracking.
- Smart Home
  - Smoke alarms, temperature sensors, light globes, and switches.



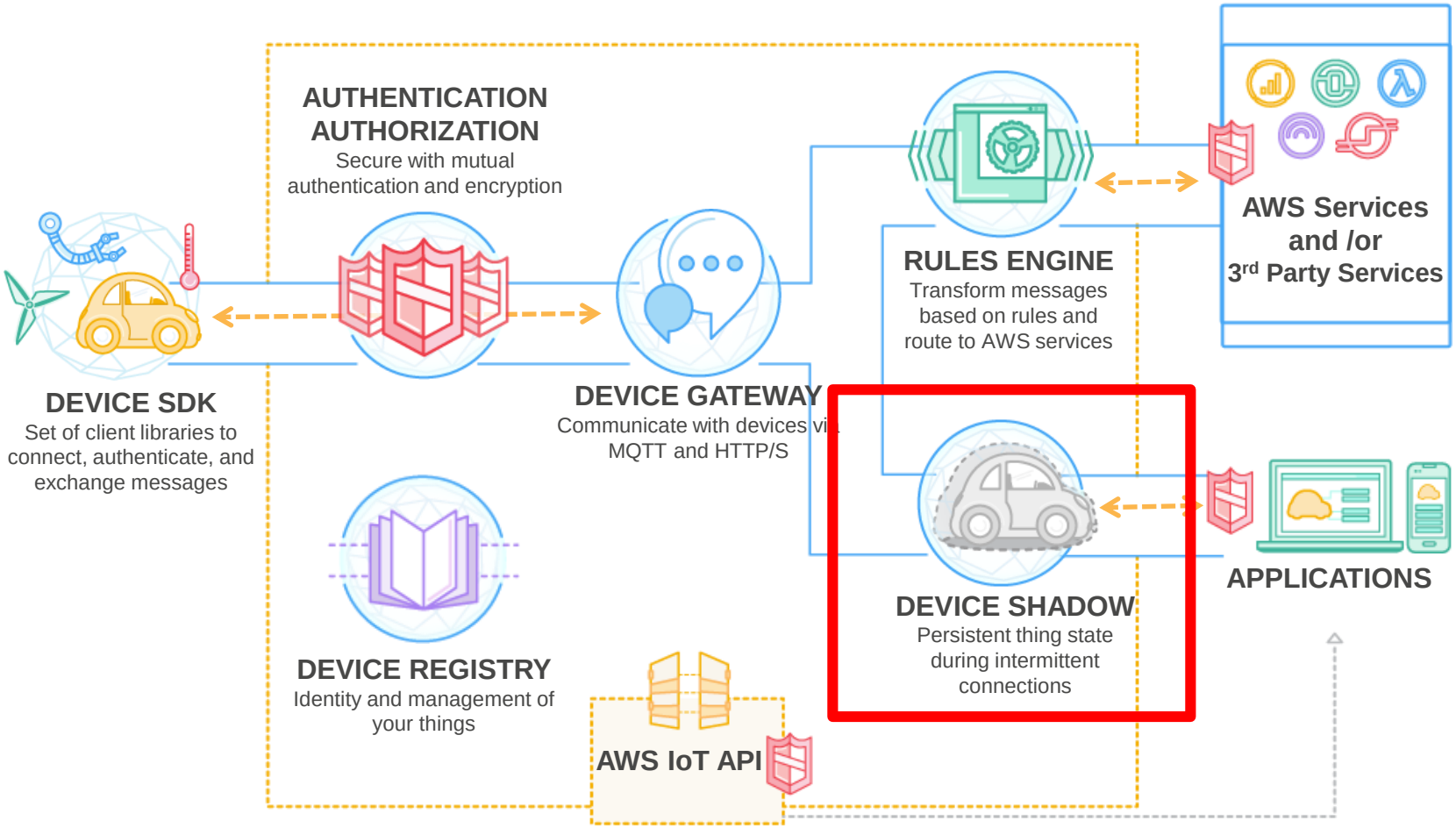
AWS IoT Framework



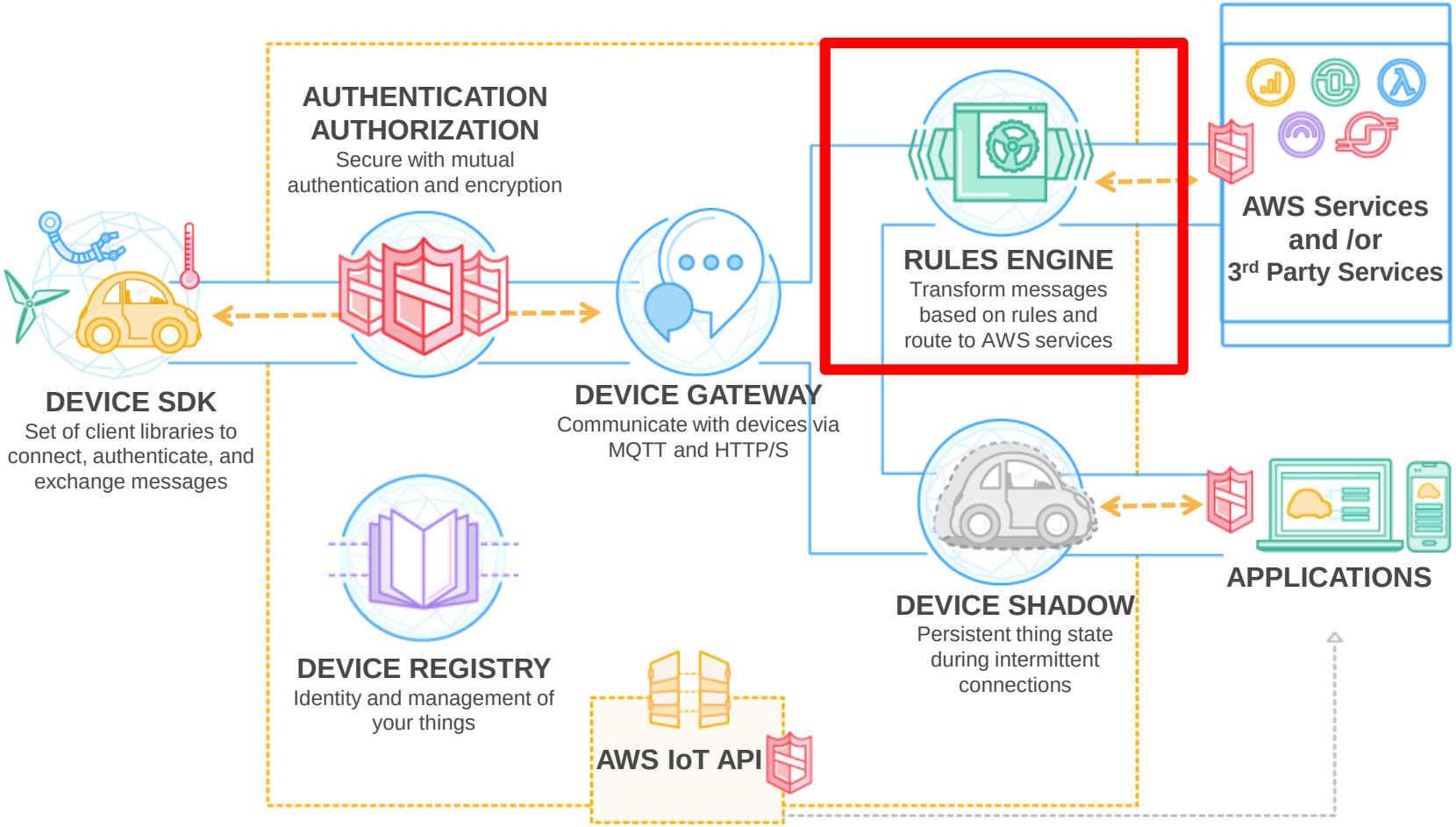
AWS IoT Framework



AWS IoT Framework



AWS IoT

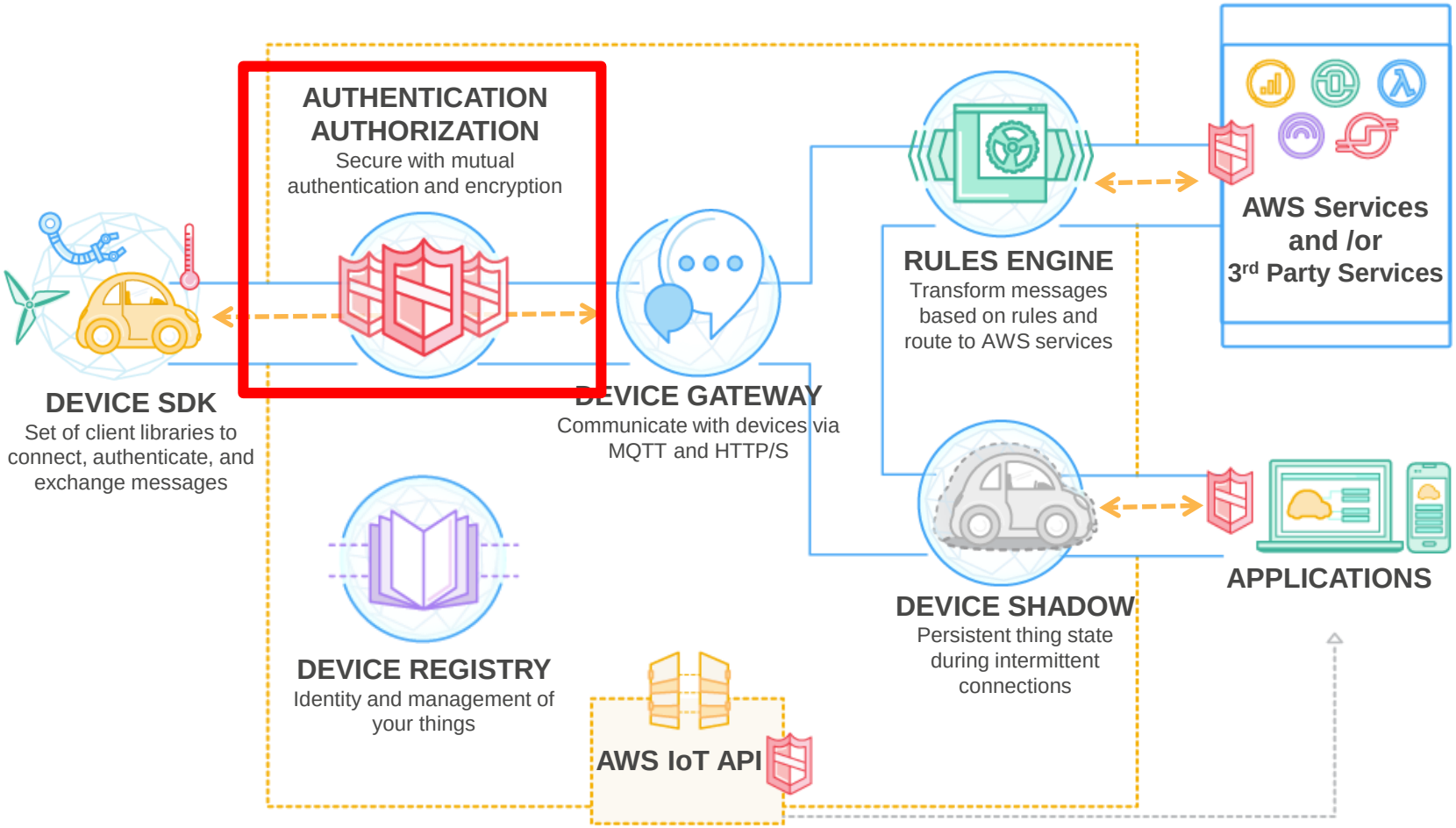

AUTHENTICATION
AUTHORIZATION
Secure with mutual authentication and encryption
RULES ENGINE
Transform messages based on rules and route to AWS services
AWS Services and /or 3rd Party Services
DEVICE SDK
Set of client libraries to connect, authenticate, and exchange messages
DEVICE GATEWAY
Communicate with devices via MQTT and HTTP/S
DEVICE SHADOW
Persistent thing state during intermittent connections
APPLICATIONS
DEVICE REGISTRY
Identity and management of your things
AWS IoT API

AWS IoT - Rules Engine

Rules give your devices the ability to interact with AWS services. Rules are analyzed and actions are performed based on the MQTT topic stream

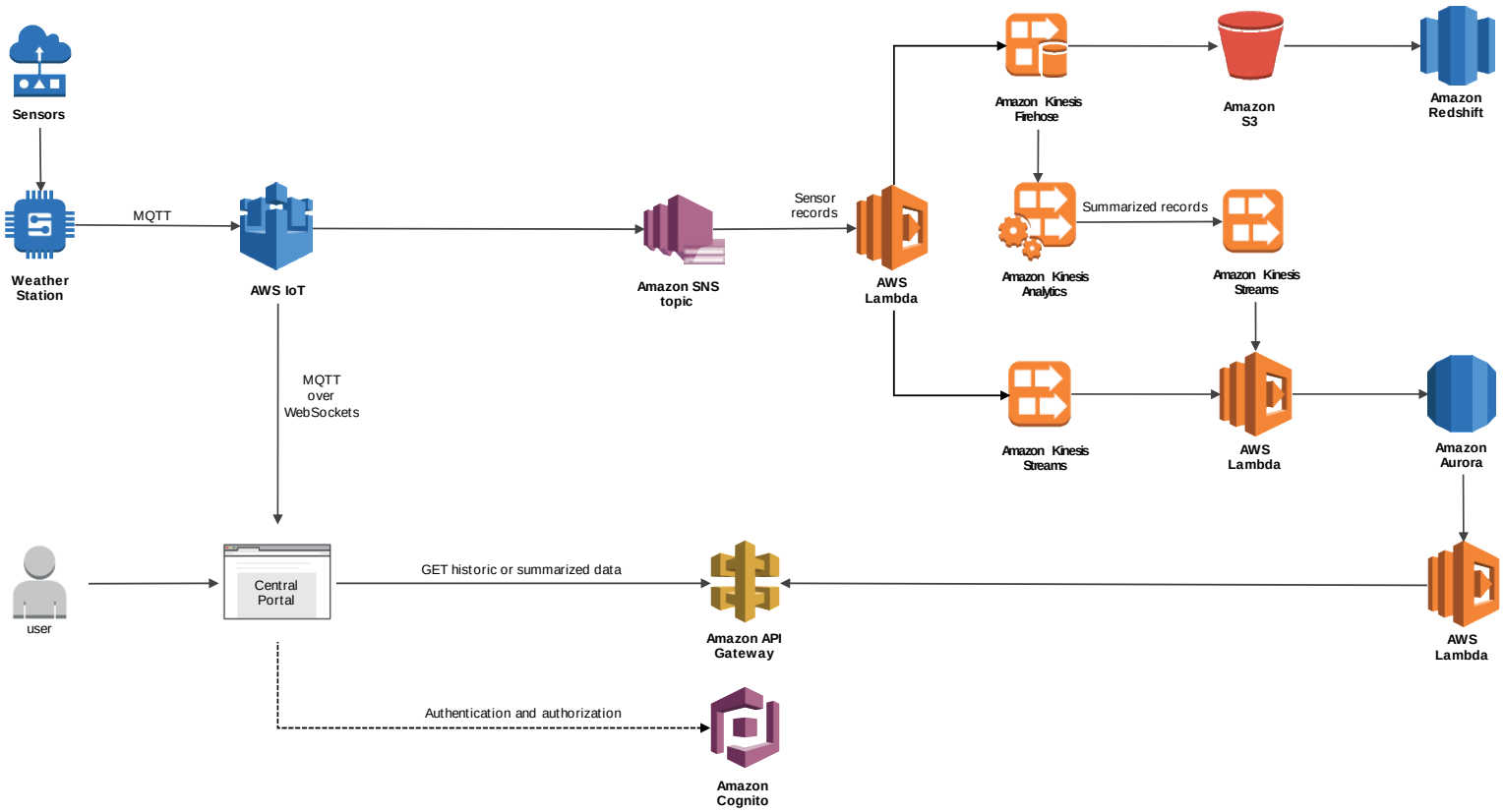
- Augment or filter data received from a device.
- Write data received to an Amazon DynamoDB database.
- Save a file to Amazon S3.
- Send a push notification to all users of Amazon SNS.
- Publish data to an Amazon SQS queue.
- Invoke a Lambda function to extract data.
- Process messages from a large number of devices using Amazon Kinesis.
- Republish the message to another MQTT topic.

AWS IoT Framework

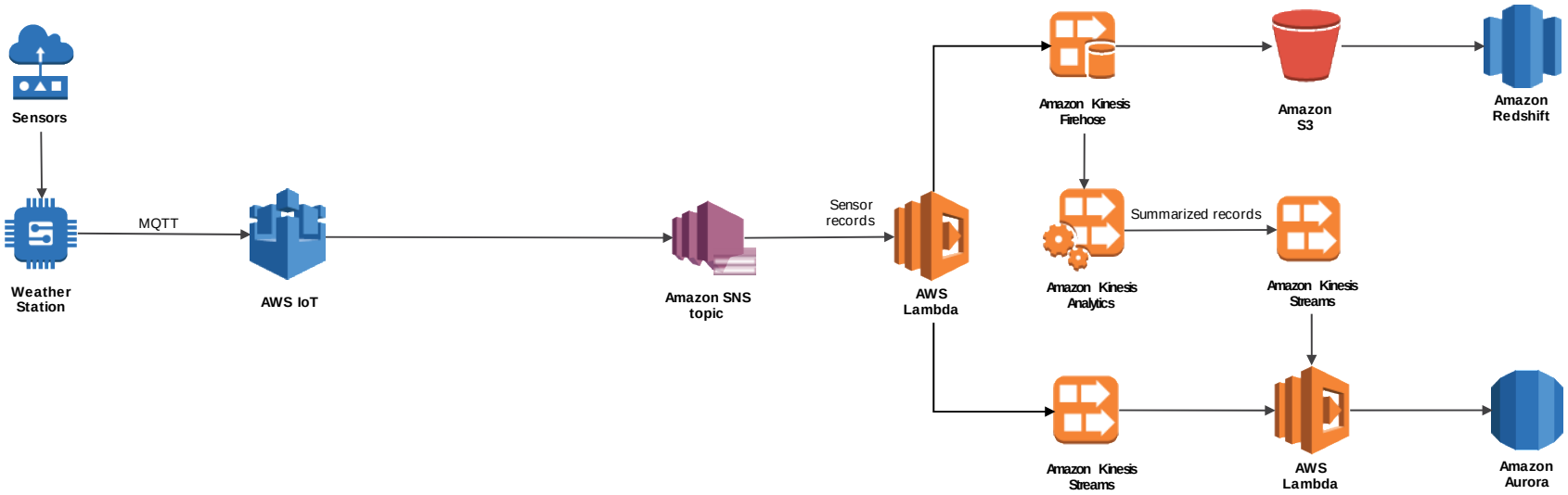


Global Weather Service Architecture

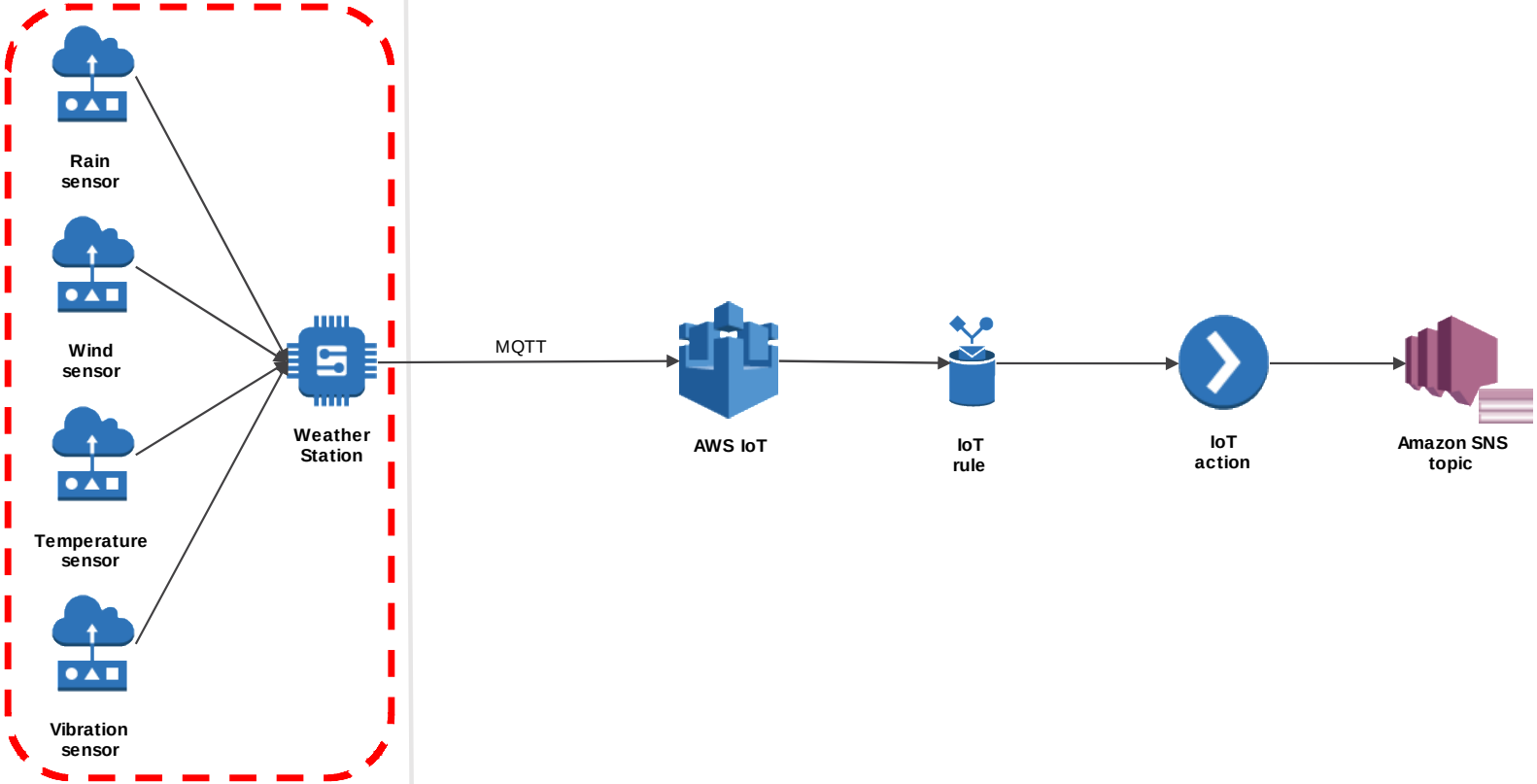
Global Weather Service Architecture



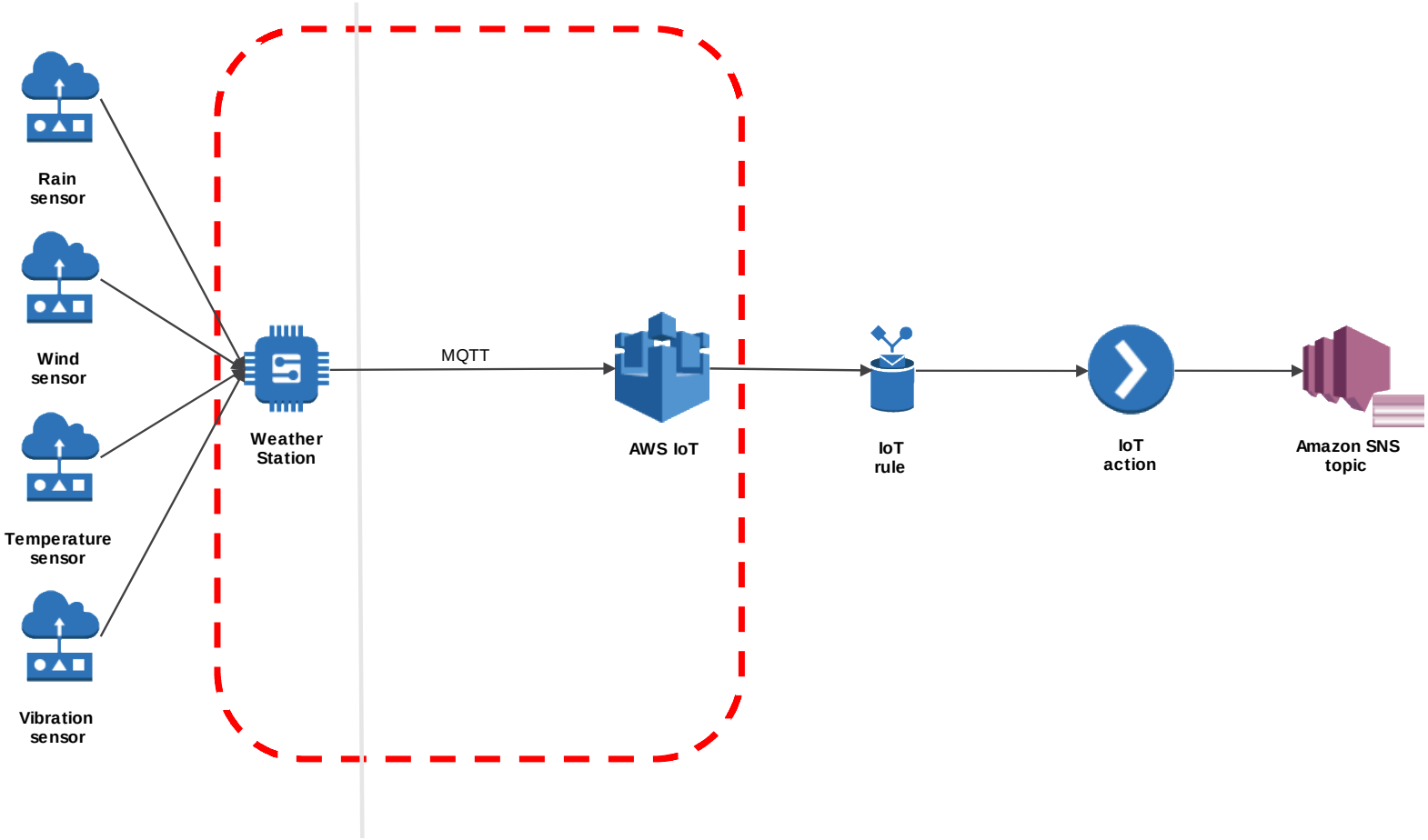
Global Weather Service Architecture



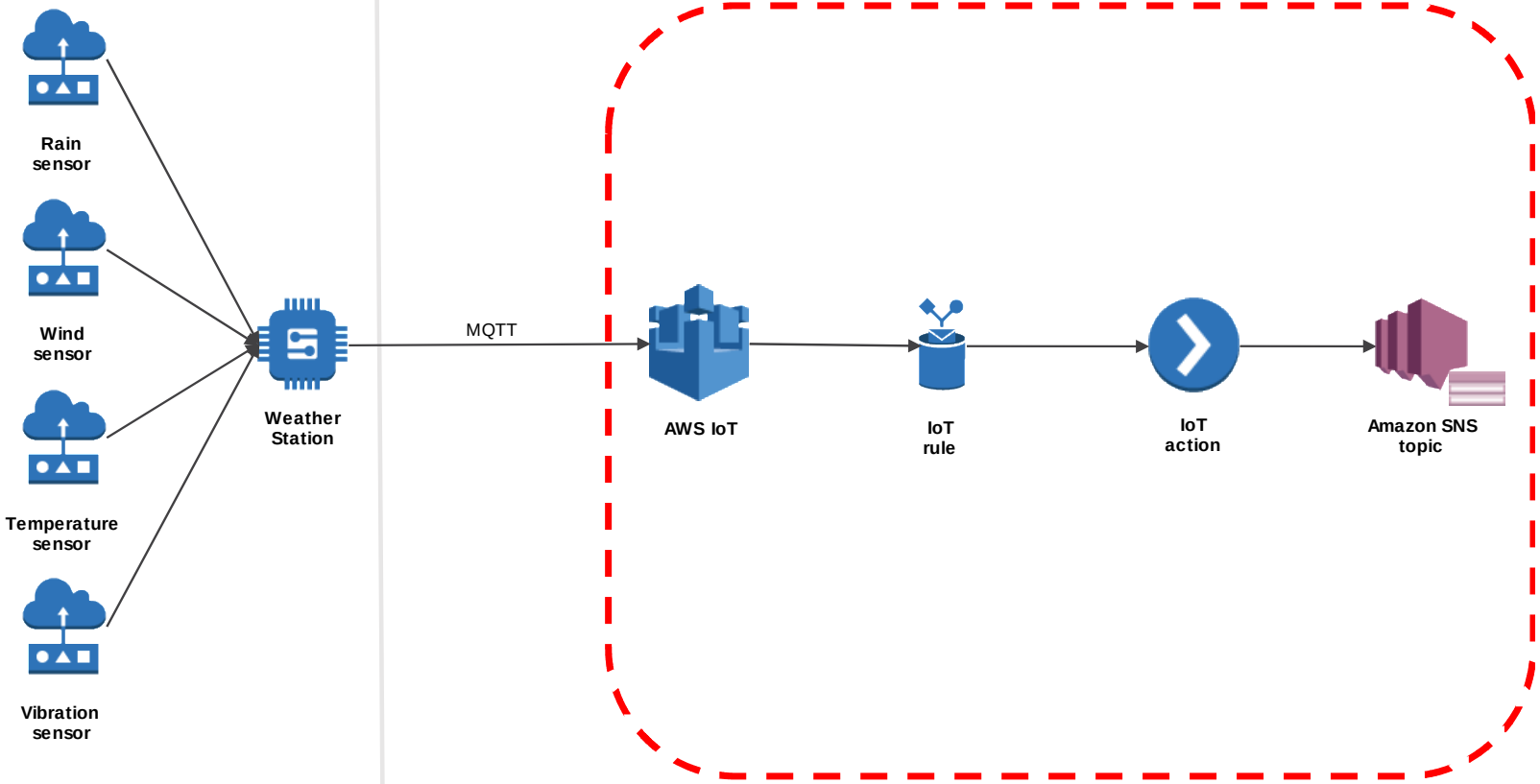
Acquisition Architecture



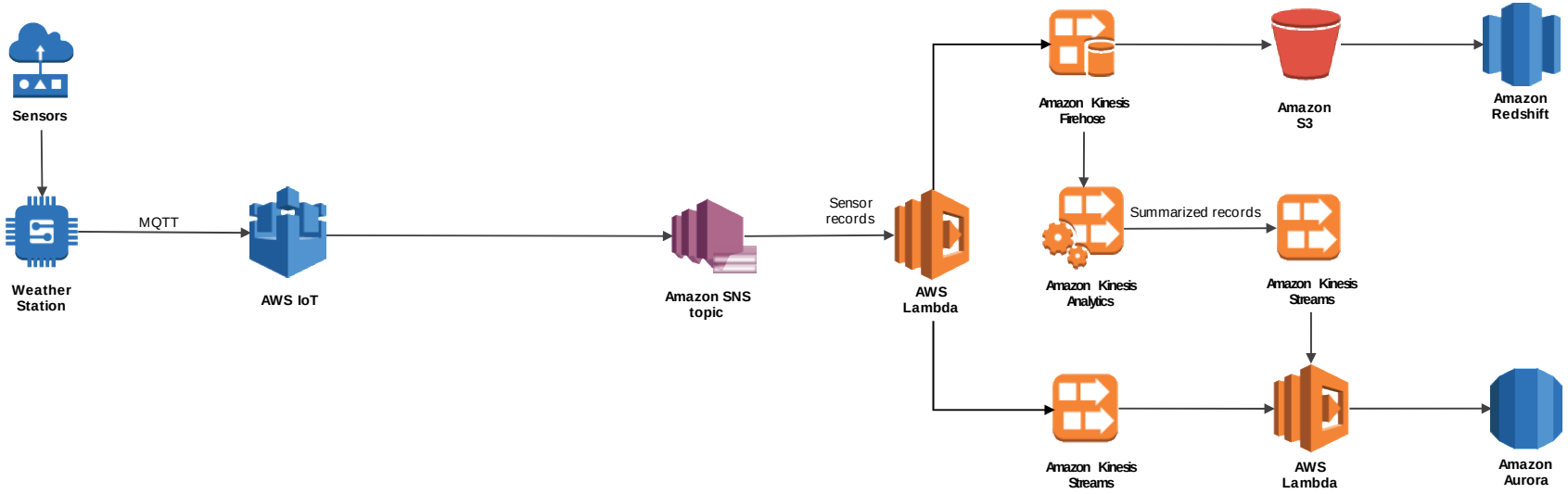
Acquisition Architecture



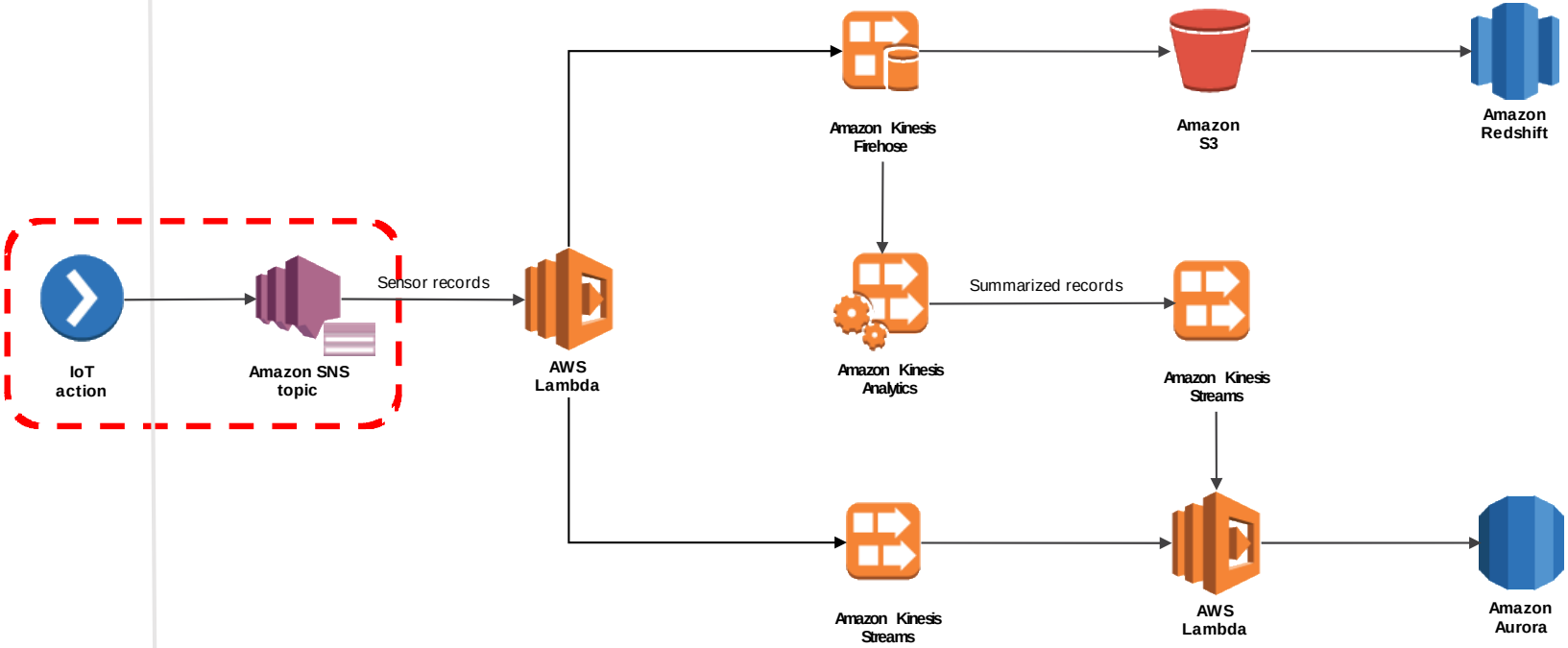
Acquisition Architecture



Global Weather Service Architecture



Processing Architecture



AWS IoT – Rule Setup

Incoming MQTT Topic structure

```
weather/<state>/<city>/<station_id>/<sensor_type>/<sensor_id>
```

AWS IoT – Rule Setup

SQL Statement

```
SELECT * FROM
topic(6) AS sensor_id, topic(4) AS station_id,
topic(5) AS sensor, sensor_timestamp,
cast(sensor_value as float) AS sensor_value,
cast(sensor_value_smoothed as float) AS sensor_value_smoothed,
cast(direction as int) AS direction
```

AWS IoT – Rule Setup

```
SELECT * FROM
topic(6) AS sensor_id, topic(4) AS station_id,
topic(5) AS sensor, sensor_timestamp,
cast(sensor_value as float) AS sensor_value,
cast(sensor_value_smoothed as float) AS sensor_value_smoothed,
cast(direction as int) AS direction
```

References the AWS IoT MQTT topic segment

<topic 1>/<topic 2>/…/<topic n>

AWS IoT – Rule Result


```
{
  "value": 0.610802791886758,
  "direction": -1,
  "smoothed": 0.98431521238906 55,
  "timestamp": 1472611226005
}
```


Incoming payload

AWS IoT – Rule Result

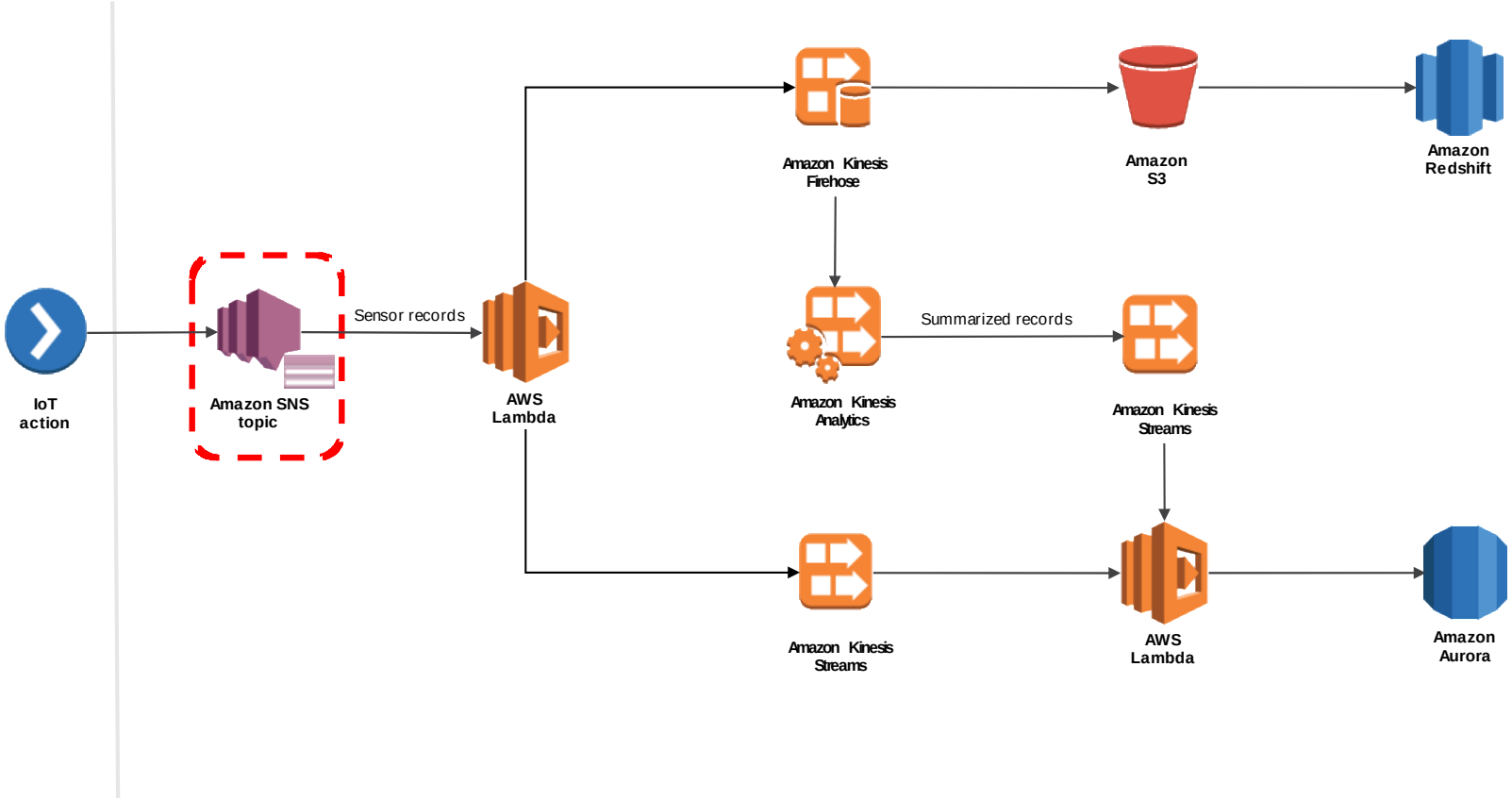
Incoming payload

```
{
  "value": 0.610802791886758,
  "direction": -1,
  "smoothed": 0.9843152123890655,
  "timestamp": 1472611226005
}
```

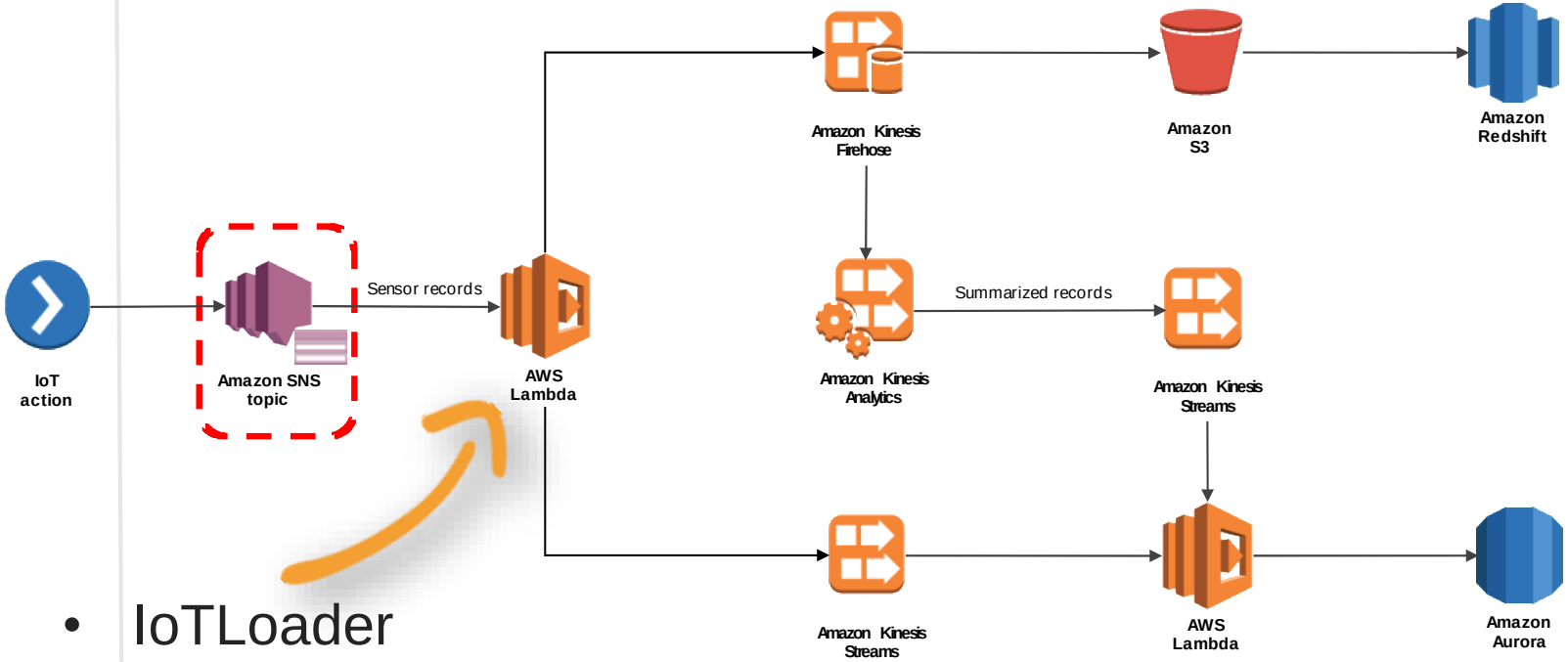
Transformed payload

```
{
  "sensor_id": "bQ7KcaMEas",
  "station_id": "vzqHb8vghO",
  "sensor": "vib",
  "timestamp": 1472611226005,
  "value": 0.610802791886758,
  "value_smoothed": 0.9843152123890655,
  "direction": -1
}
```

Processing Architecture

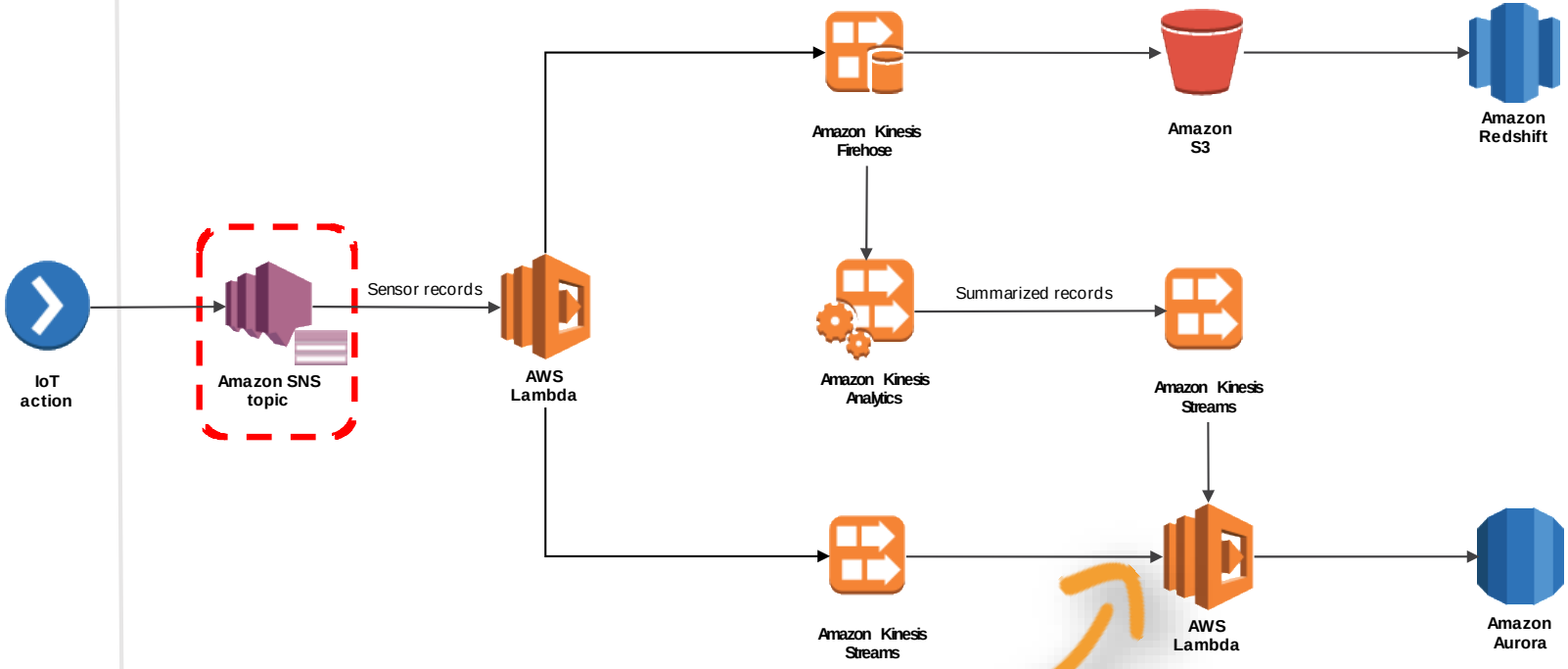


Processing Architecture



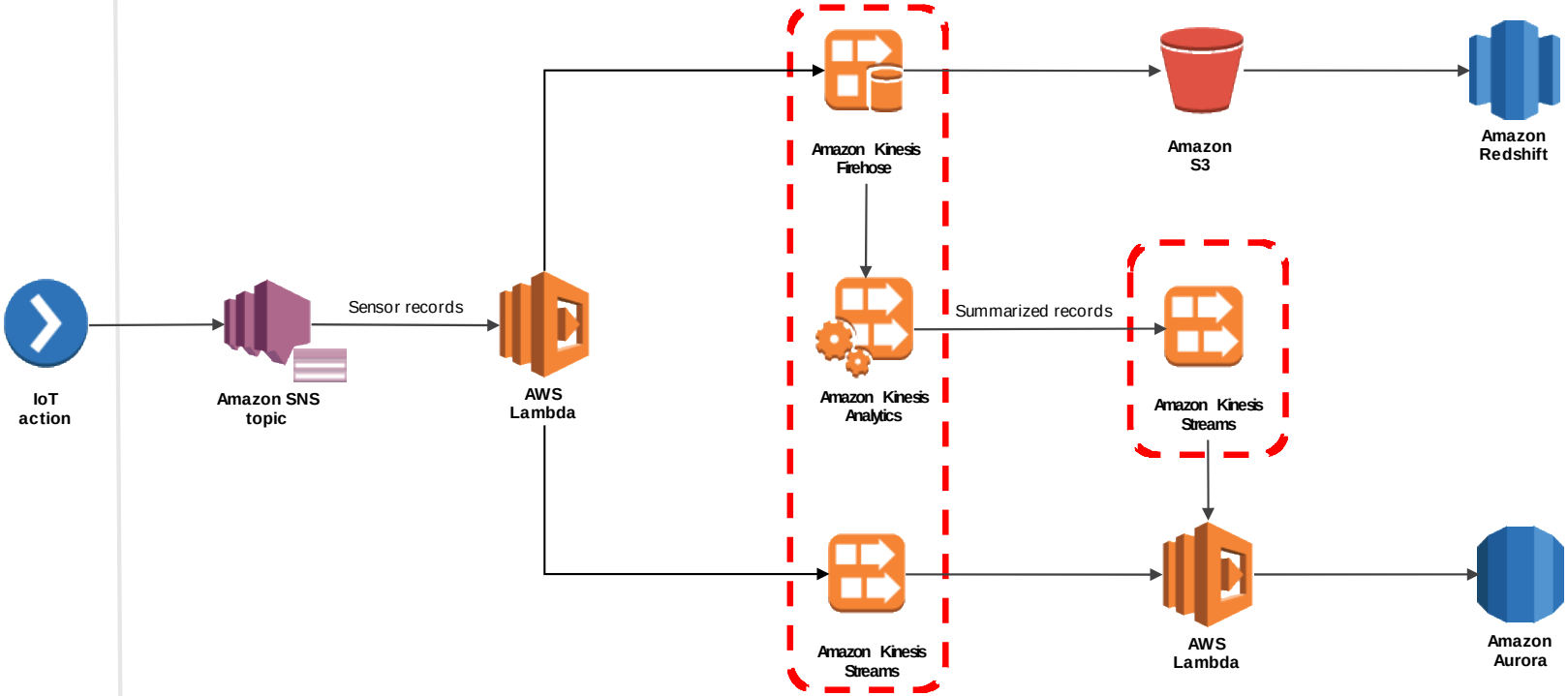
- IoTLoader
  - Process sensor data records from an AWS IoT action and injects them into an Amazon Kinesis stream and Amazon Kinesis Firehose delivery stream.

Processing Architecture



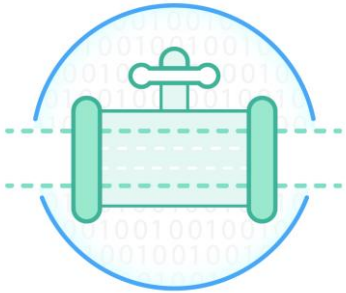
- RdsLoader
  - Process sensor data records from an Amazon Kinesis stream and inserts them into an Amazon Aurora RDS database.

Processing Architecture



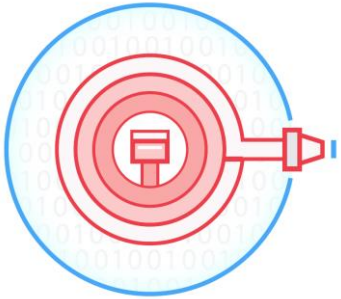
Amazon Kinesis: Streaming Data Made Easy

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



Amazon Kinesis Streams

- For technical developers
- Build your own custom applications that process or analyze streaming data



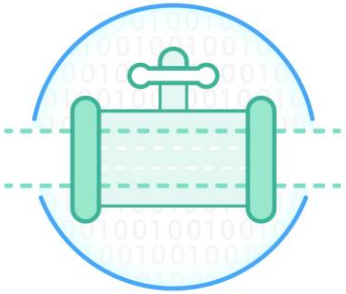
Amazon Kinesis Firehose

- For ETL, data engineer
- Easily load massive volumes of streaming data into S3, Amazon Redshift and Amazon Elasticsearch Service

Amazon Kinesis Analytics

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

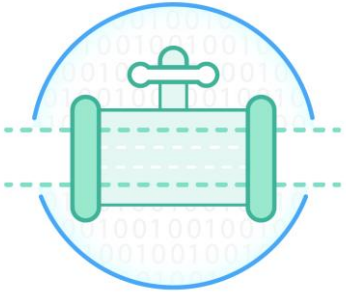
Amazon Kinesis - Streaming Data Made Easy



Amazon Kinesis Streams

Low latency streaming ingest at scale

Amazon Kinesis - Streaming Data Made Easy



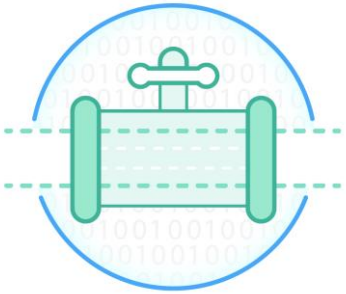
Amazon Kinesis Streams

Low latency streaming ingest at scale

Amazon Kinesis Analytics

Streaming analytics in near real-time

Amazon Kinesis - Streaming Data Made Easy

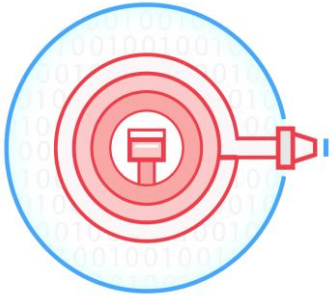


Amazon Kinesis Streams

Low latency streaming ingest at scale

Amazon Kinesis Analytics

Streaming analytics in near real-time



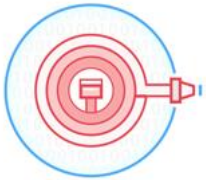
Amazon Kinesis Firehose

Batch data delivery based on time/size into S3

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 Service** and a data latency of 60 seconds or higher.

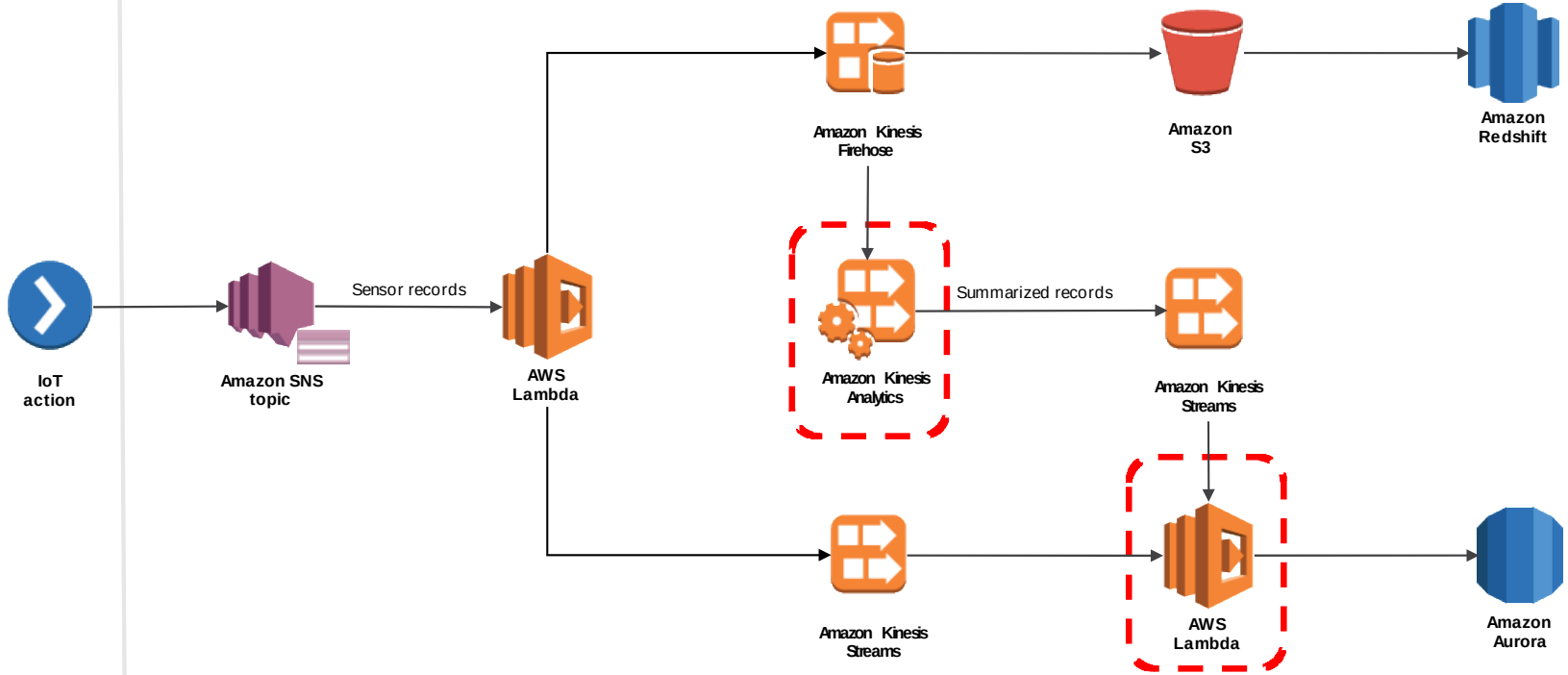
Use SQL To Build Real-Time Applications

Connect to streaming source

Easily write SQL code to process streaming data

Continuously deliver SQL results

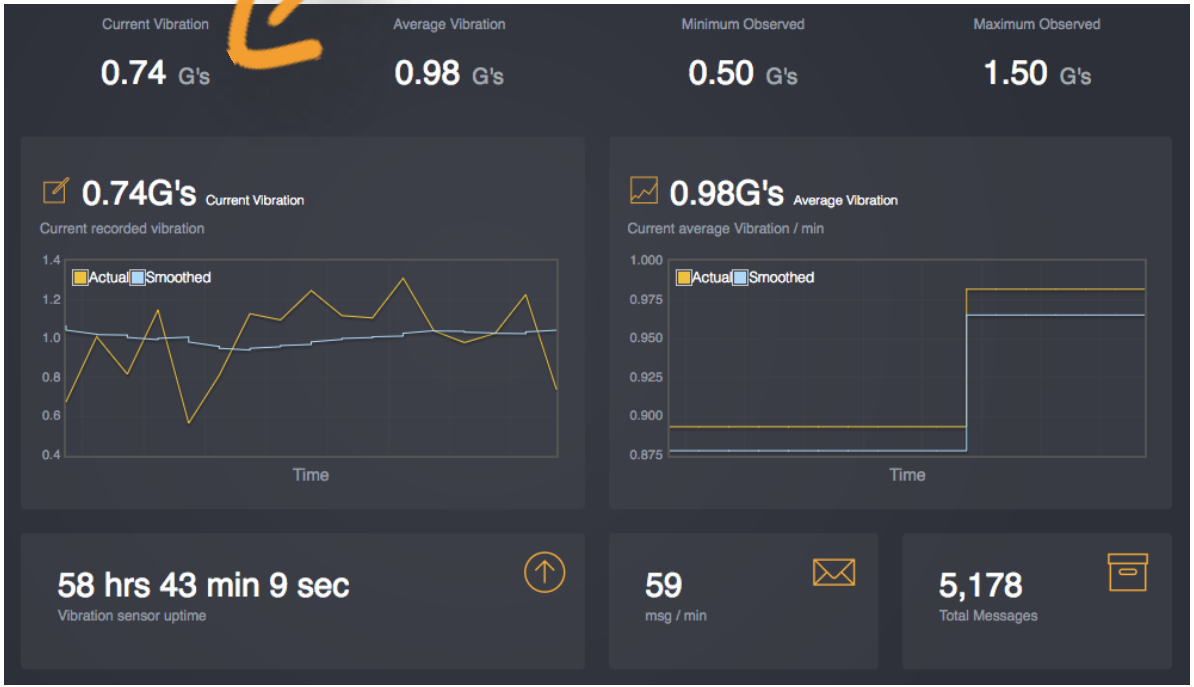
Processing Architecture



Amazon Kinesis Analytics – Answering Questions

Amazon Kinesis Analytics – Answering Questions

What is the current value ?



Amazon Kinesis Analytics – Answering Questions

What is the average value ?

Amazon Kinesis Analytics – Answering Questions

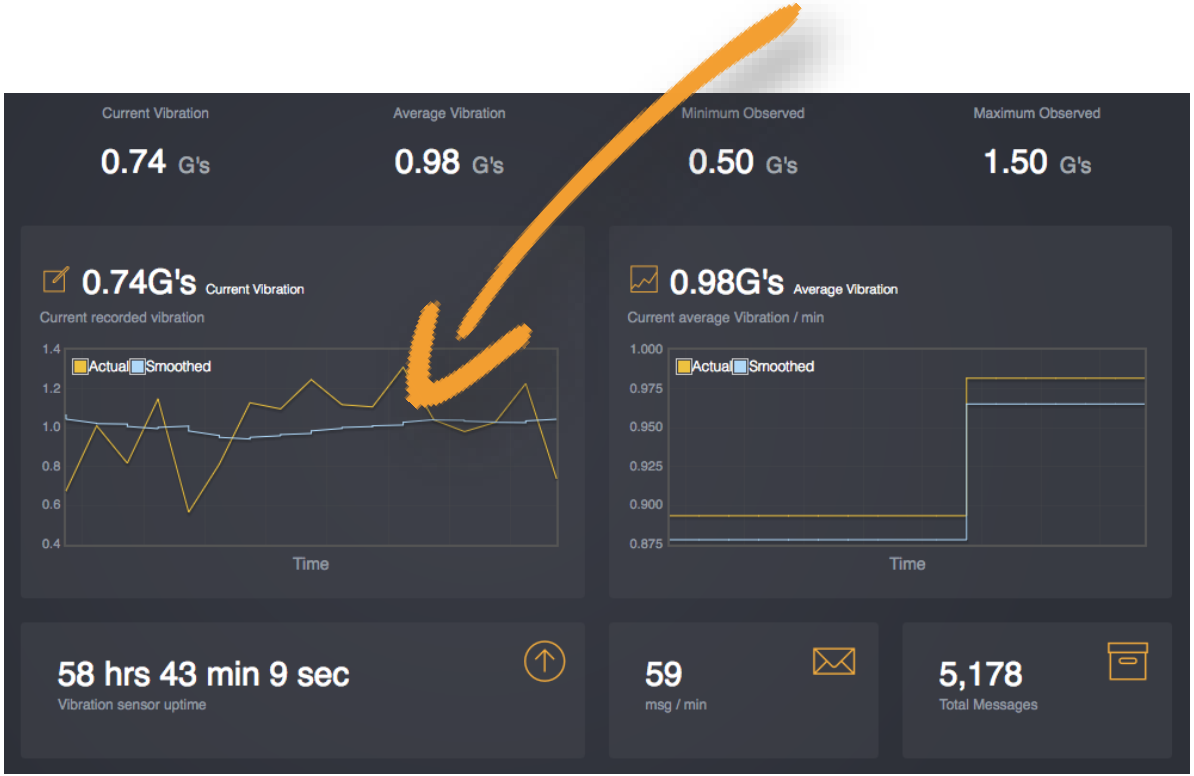
What is the minimum value ?

Amazon Kinesis Analytics – Answering Questions

What is the maximum value ?

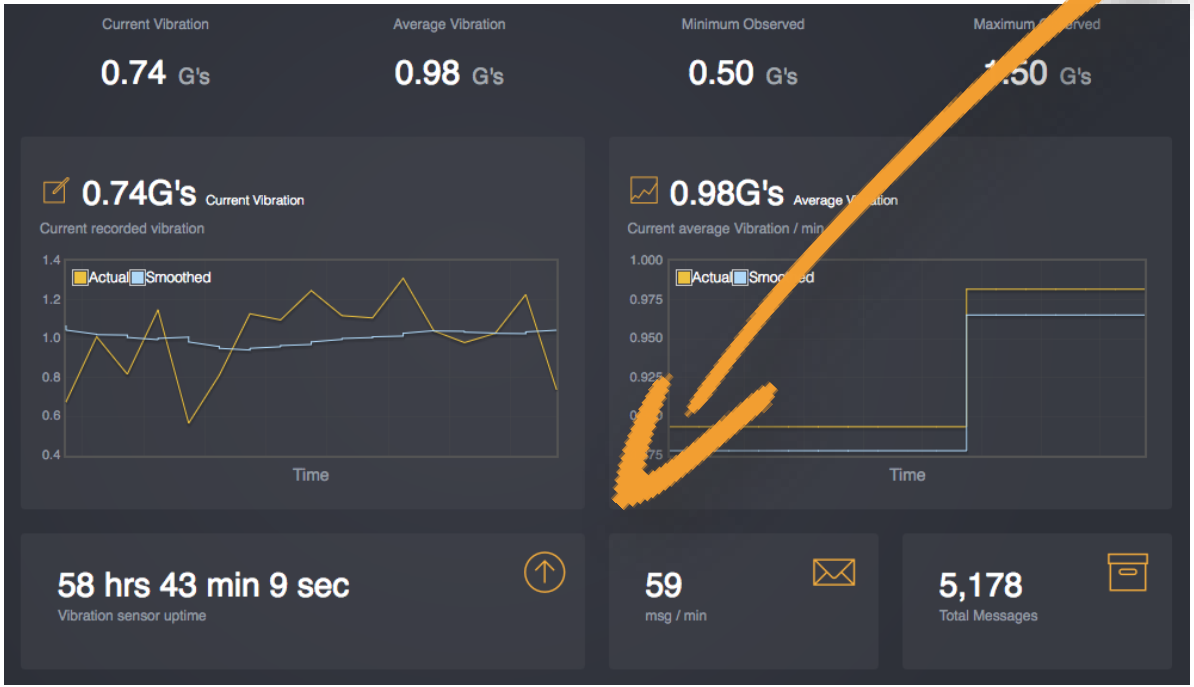
Amazon Kinesis Analytics – Answering Questions

Visual graphs for short term trending



Amazon Kinesis Analytics – Answering Questions

Service performance statistics



Amazon Kinesis Analytics – Processing Setup

```sql
CREATE OR REPLACE STREAM "DESTINATION_SQL_STREAM" ("sensor_id" VARCHAR(32), "sensor" VARCHAR(15),
    "station_id" VARCHAR(32), "sensor_avg_value" double, "sensor_smooth_avg_value" double,
    "60sec_sum_of_sensor_value" double, "60sec_number_of_msg" int, "record_timestamp" TIMESTAMP);

CREATE OR REPLACE PUMP "STREAM_PUMP" AS INSERT INTO "DESTINATION_SQL_STREAM"
SELECT STREAM "sensor_id", "sensor", "station_id",
    AVG("sensor_value"), AVG("sensor_value_smoothed"),
    SUM("sensor_value") AS "60sec_sum_of_sensor_value",
    count(*) AS "60sec_number_of_msg", ROWTIME AS "record_timestamp"
FROM "SOURCE_SQL_STREAM_001"
GROUP BY "sensor_id","sensor","station_id",
    FLOOR(("SOURCE_SQL_STREAM_001".ROWTIME - TIMESTAMP '1970-01-01 00:00:00') SECOND / 60 TO SECOND);
```

Cancel

Source data | Real-time analytics | Destination | Applicati

In-application streams:

DESTINATION_SQL_STREAM

error_stream

Pause results New results will be added every 2-10 seconds

Scroll to bottom when new results arrive.

Filter by column name

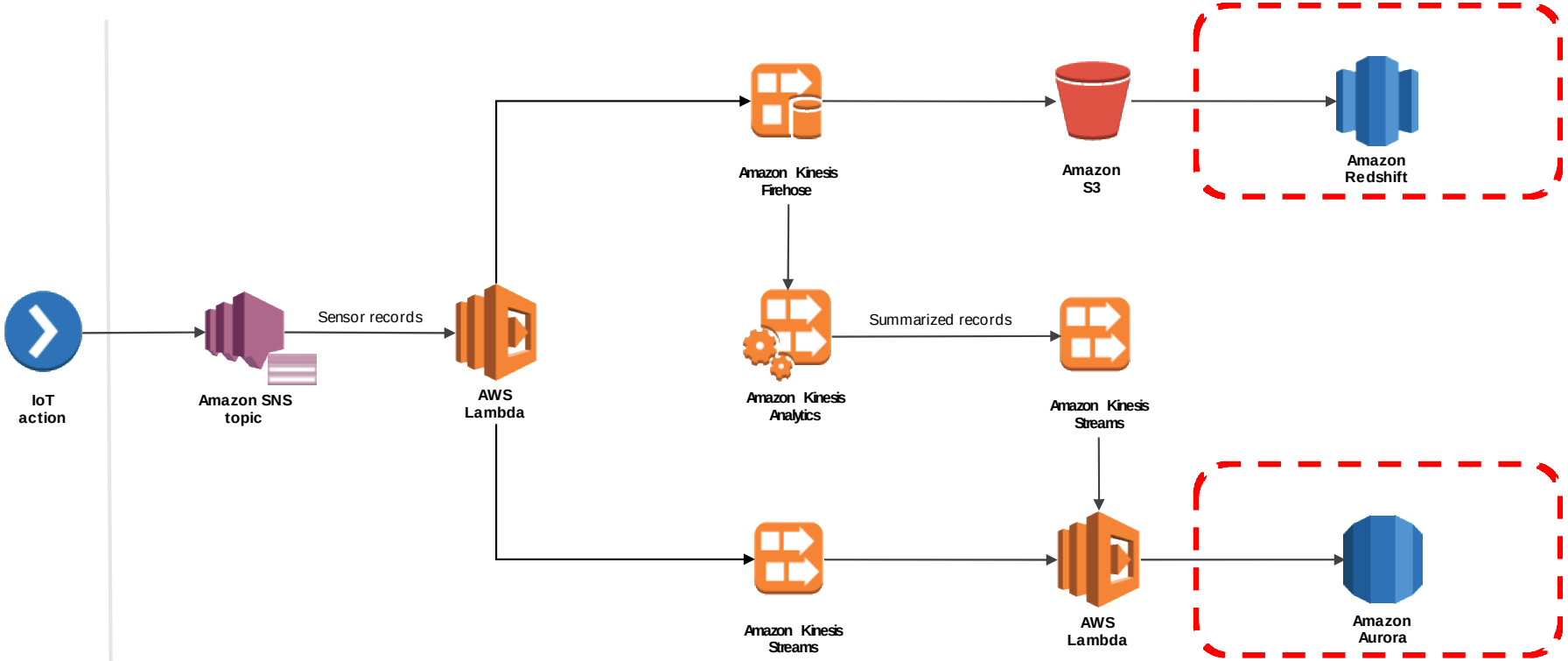
ROWTIME	sensor_id	sensor	station_id	sensor_avg_value	sensor_sm
2016-11-11 01:44:00.0	813c174428fc4843	temp	qwbKAMlbZW	63.23000000000001	63.2300000
2016-11-11 01:44:00.0	caaed84a552deebf	rain	qwbKAMlbZW	0.0	0.0

Amazon Kinesis Analytics – Processing Result

Emitted payload


```
{
  "sensor_id": "dc2b8383eb79fe49",
  "sensor": "vib",
  "station_id": "qwbKAMlbZW",
  "sensor_avg_value": 1.072153418386984,
  "sensor_smooth_avg_value": 1.0158438044679172,
  "60sec_sum_of_sensor_value": 64.32920510321904,
  "60sec_number_of_msg": 60,
  "record_timestamp": "2016-11-09 06:29:00.0"
}
```

Processing Architecture



Data Store Summary

Amazon S3

- Raw long term storage for warm data
- Lifecycle management
- Reprocess and reload data

Data Store Summary

Amazon S3

- Raw long term storage for warm data
- Lifecycle management
- Reprocess and reload data

Amazon Redshift

- Optimized for data warehousing and analytics
- Query large amounts of data fast
- Scale to increase performance

Data Store Summary

Amazon S3

- Raw long term storage for warm data
- Lifecycle management
- Reprocess and reload data

Amazon Redshift

- Optimized for data warehousing and analytics
- Query large amounts of data fast
- Scale to increase performance

Amazon Aurora

- Optimized for distributed data access
- Scale read throughput
- Fault tolerant

AWS re:Invent

Tim Bart

CTO, Hello

amazon web services

What we do

Our mission is to help people to live better through understanding themselves and the world around them.

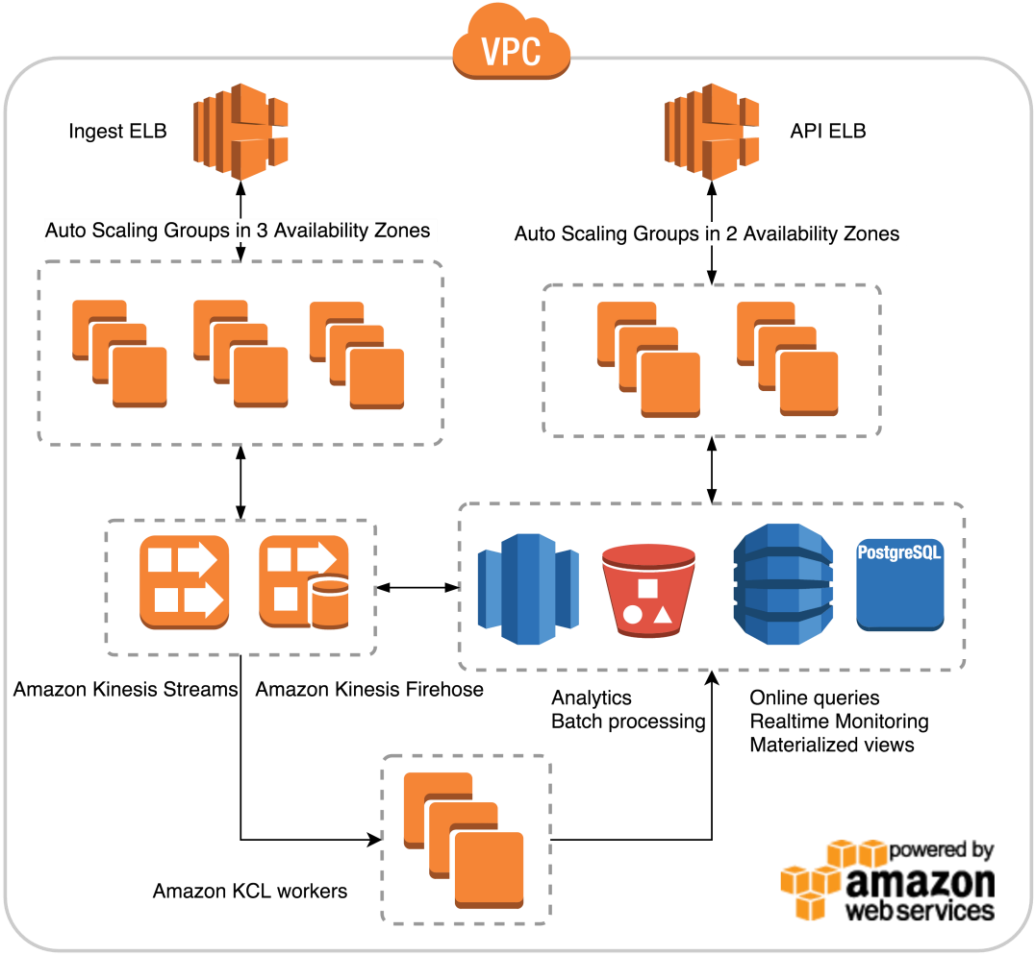
To achieve that, we build delightful products with hardware, software and data science.

Amazon Kinesis for IoT data at Hello

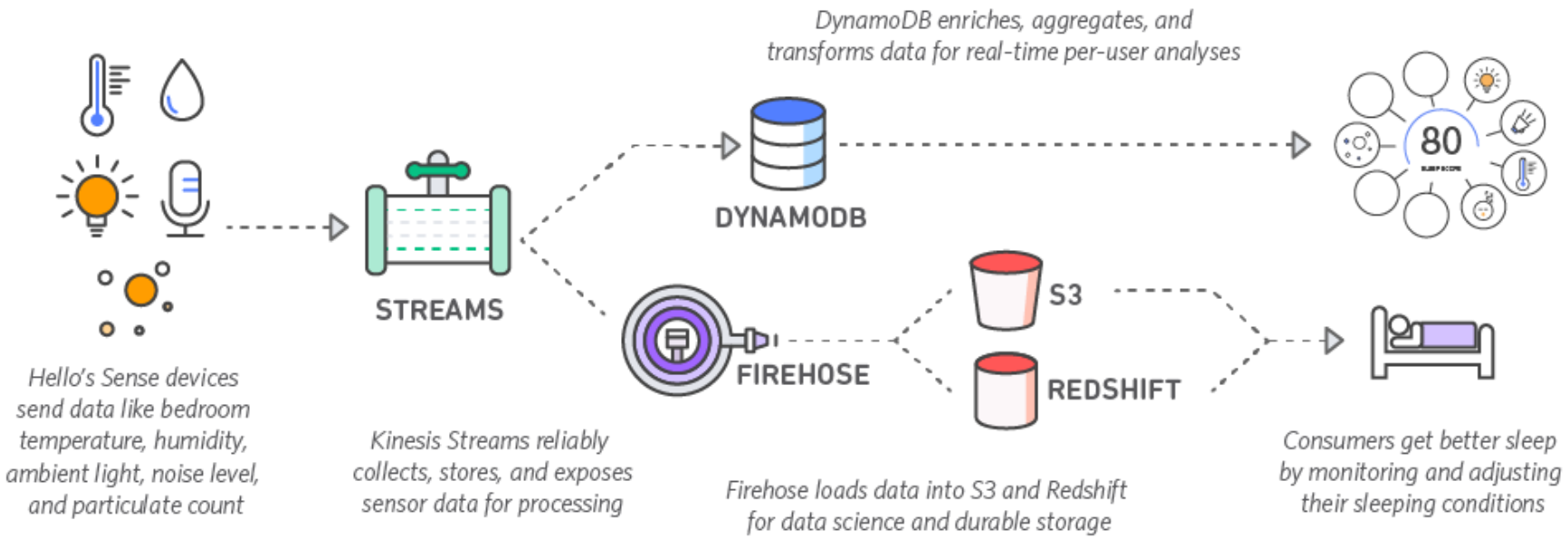
High Level View

100% of the data generated by our devices goes through Amazon Kinesis streams.

This includes sensor data, device diagnostic logs, device system metrics.



Using both Amazon Kinesis Streams & Amazon Kinesis Firehose



Why we chose Amazon Kinesis

1. Durability

2. Immutability

3. Real-time processing

4. Cost effective and very low operations overhead.

Durability

1. Many small messages (< 500 bytes) or fewer larger messages (~50kb) depending on the nature of the data.

2. Synchronous PutRecord calls to Amazon Kinesis Streams for Sensor Data. Low latency, Low throughput

3. Diagnostic data, logs, can be sent in batches as durability concerns are not as strict as sensor data. Higher latency, Higher throughput.

4. At least once delivery. Handle duplicate records by having using idempotent operations downstream. 7 days data retention.

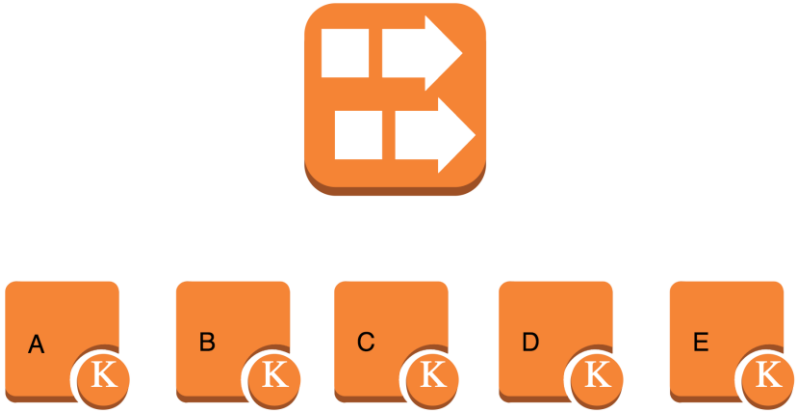
Immutability

1. Few streams, many consumers.
   ~1:10 stream/consumer

2. Experiment with AWS Lambda
   without changing anything to your
   current architecture.

3. Reprocessing all data to safely
   experiment with different algorithms.

   Run version A, B, C of your algorithm in parallel
   or update algorithm and reprocess
   all data from the stream and compare the results.



Real-time monitoring use case

Quick intro to the Amazon Kinesis Client Library

```
public interface IRecordProcessor {
    // Invoked by the KCL before data records are delivered
    // to the RecordProcessor instance
    void initialize(InitializationInput initializationInput);

    //Process data records. The KCL will invoke this method to deliver data records
    // to the application.
    void processRecords(ProcessRecordsInput processRecordsInput);

    //Invoked by the Amazon Kinesis Client Library to indicate it
    // will no longer send data records to this
    void shutdown(ShutdownInput shutdownInput);
}
```

Track last seen time for each device

```
// LastUploadProcessor implements IRecordProcessor

Jedis jedis = new Jedis(host, port); // elasticache host + port
Pipeline pipeline = jedis.pipelined();

for( Record record : records) {
    SensorData sensorData = parseFrom( record )
    pipeline.zadd(LAST_SEEN_KEY, sensorData.id(), sensorData.unix());
    pipeline.exec();
}
```

Lessons learned

- Use the same stream for data archival & analytics.
- Split your streams in multiple shards early.
- The Amazon Kinesis Client Library (KCL) makes writing consumers really easy. Use Auto Scaling groups for automatic failover or use AWS Lambda and don’t worry about it.
- Many independent consumers let you experiment and deploy safely.

Lessons learned

- Choose your serialization protocol wisely.
- Use Amazon Kinesis Analytics if you serialization protocol is CSV or JSON.
- You will likely have to work around the 5 reads/shard/second limitation

AWS Lambda fanout

Use AWS Lambda to fan out Amazon Kinesis Streams to most AWS services.

https://github.com/awslabs/aws-lambda-fanout

Summary

IoT with real-time analytics provides meaningful information, not just data

Scale without intervention or cost

Remove management and scaling overhead to accelerate innovation

Thank you!

Remember to complete your evaluations!