Amazon Kinesis makes it easy to collect, process, and analyze real-time, streaming data so you can get timely insights and react quickly to new information. In this session, we present an end-to-end streaming data solution using Kinesis Streams for data ingestion, Kinesis Analytics for real-time processing, and Kinesis Firehose for persistence. We review in detail how to write SQL queries using streaming data and discuss best practices to optimize and monitor your Kinesis Analytics applications. Lastly, we discuss how to estimate the cost of the entire system.

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

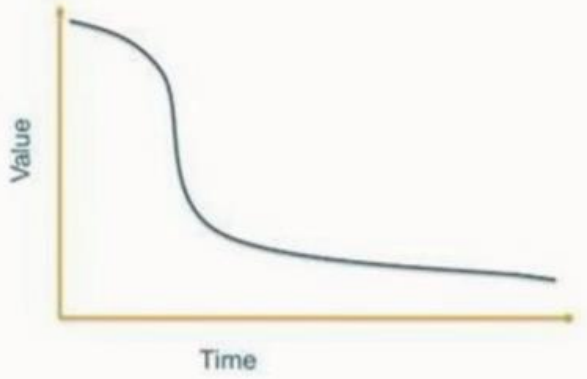
With stream processing is about continuous data capture, you are getting smaller events files like 1kb log files that are going to be aggregated and them written to a persistence store location like S3.

Why? Data loses value over time

Ingest data as it is generated

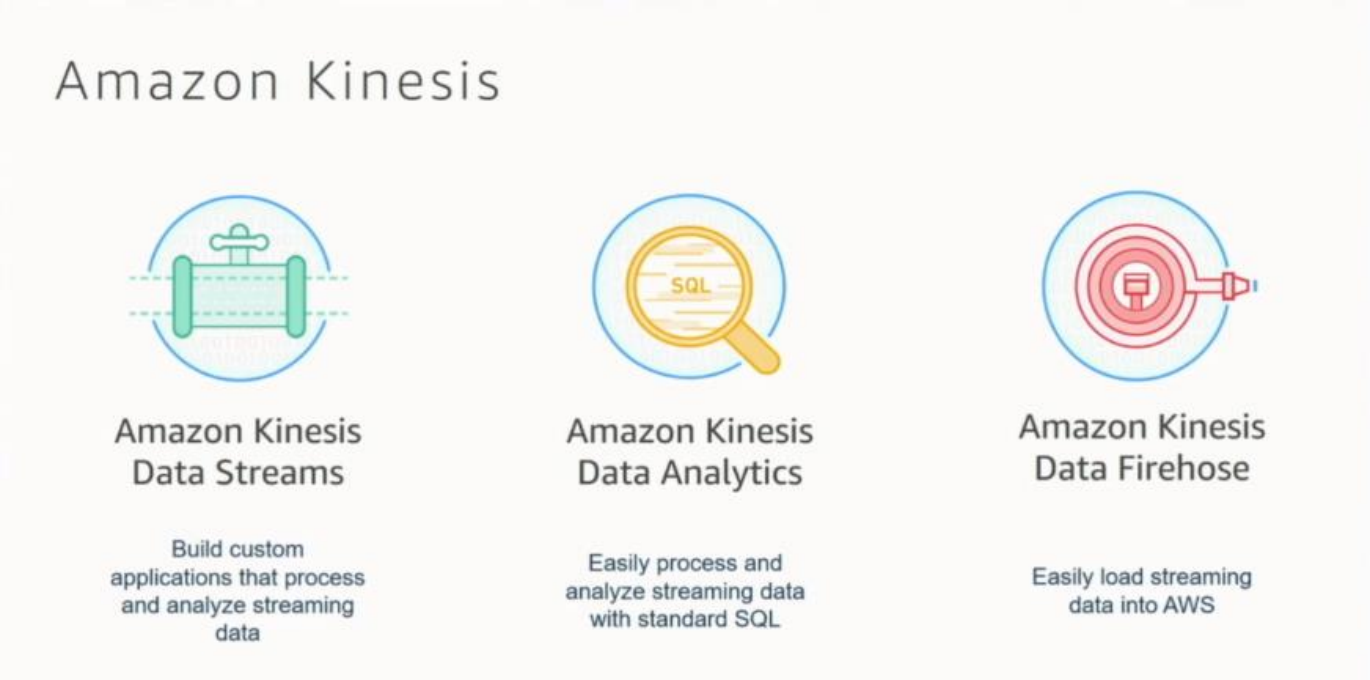
Analyze data in real time to get insights immediately

Deliver data to in seconds instead of hours

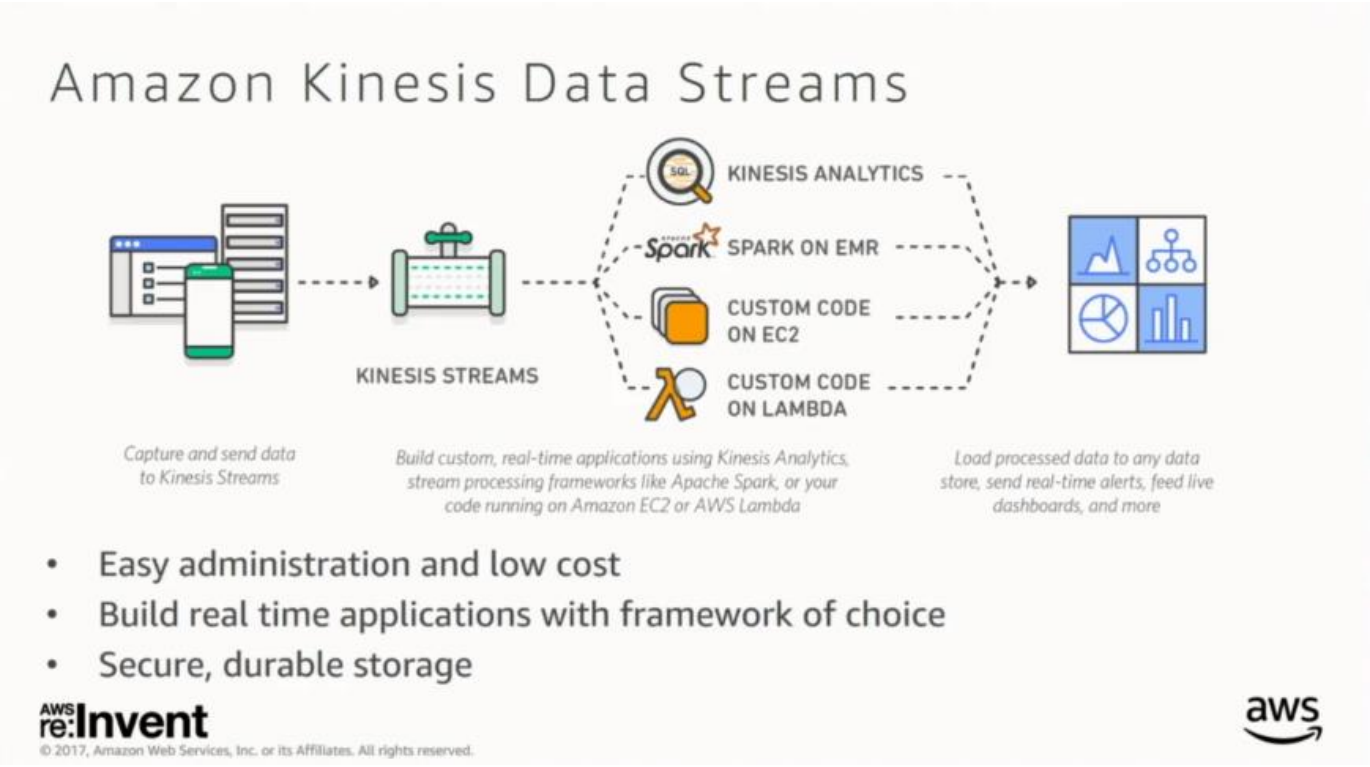


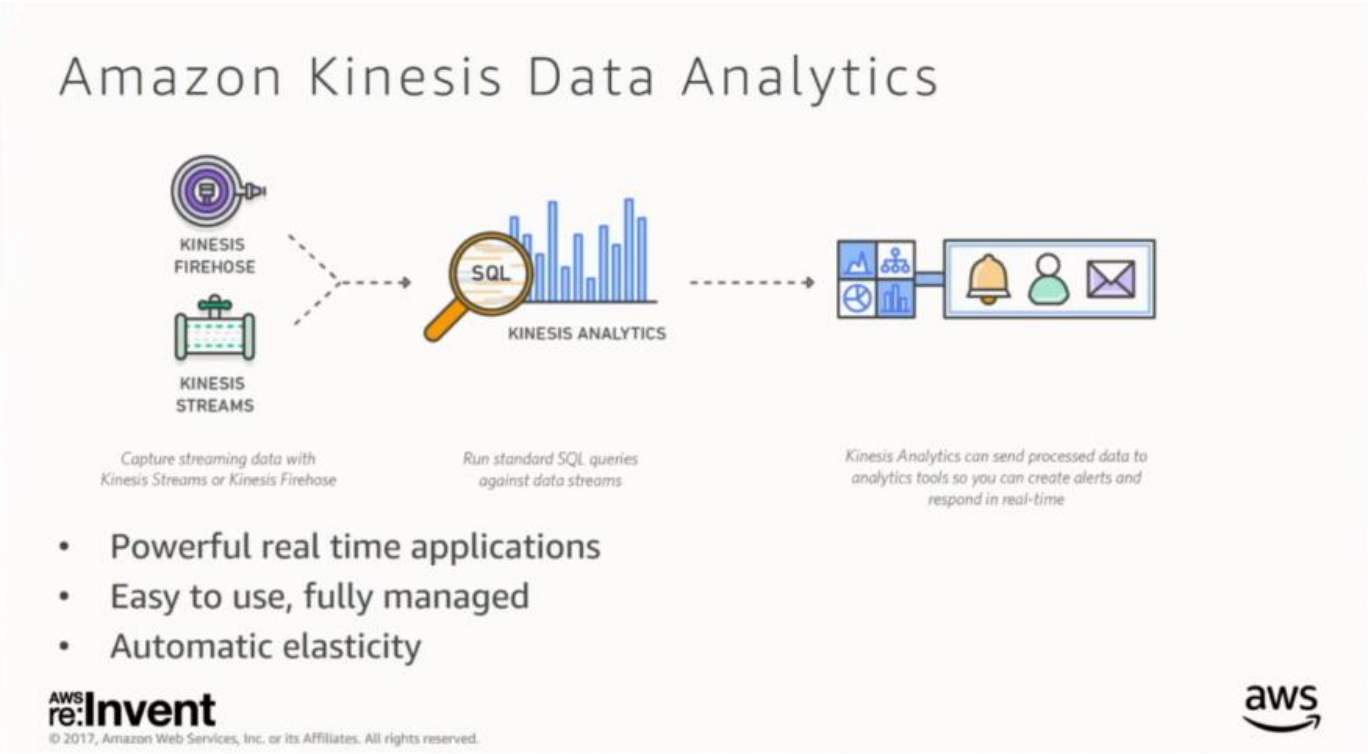
Simple Pattern for Streaming Data

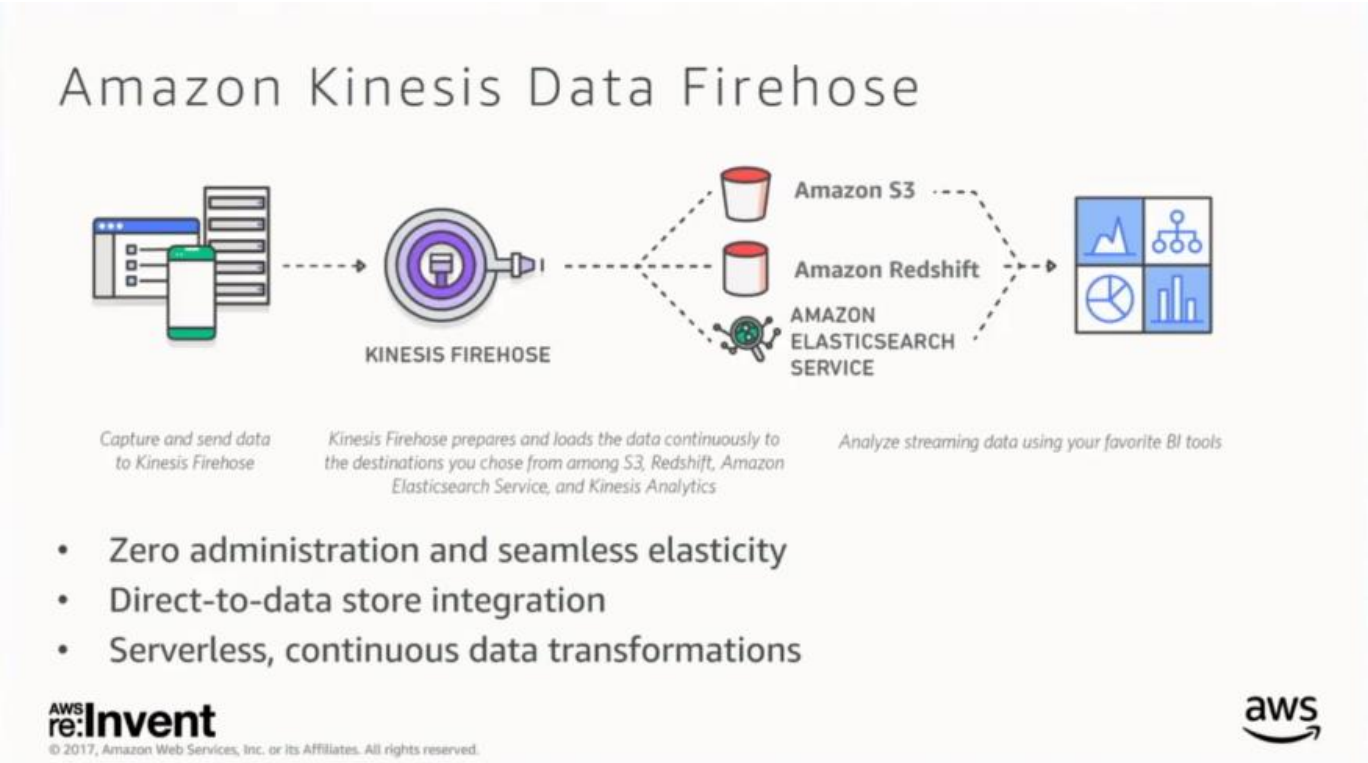
Data Producer	Streaming Service	Data Consumer
Continuously creates data	Durably stores data	Continuously processes data
Continuously writes data to a stream	Provides temporary buffer that preps data	Cleans, prepares, & aggregates
Can be almost anything	Supports very high-throughput	Transforms data to information
Mobile Client	Amazon Kinesis	Amazon Kinesis app



You will build a custom data producer that writes to the Kinesis Streams continuously using PUT. You then write data consumers to consume data off the stream. Firehose allows you to create a delivery stream that stores the data stream stores like S3 durably.







When you use Kinesis Data Firehose you create a delivery stream, after you create the delivery stream all you worry about is now writing data to the delivery stream. The data is then delivered without any other service to the destination of your choice like an S3 bucket. When you configure a Firehose you have to configure some different things, the first is the ***Buffering Hints*** which are a time period and a size of how large the data size or how often you want to deliver data to your destination like S3 e.g. you will like to deliver your data in 100mb files to S3 bucket every 10 mins. Saving large S3 files can help optimize your batch processing workloads. You can also do encryption, compression, inline transformations on the data via lambda for a serverless ETL pipeline before having Firehose put the data into S3.

Amazon Kinesis Data Analytics Applications

Connect to streaming source

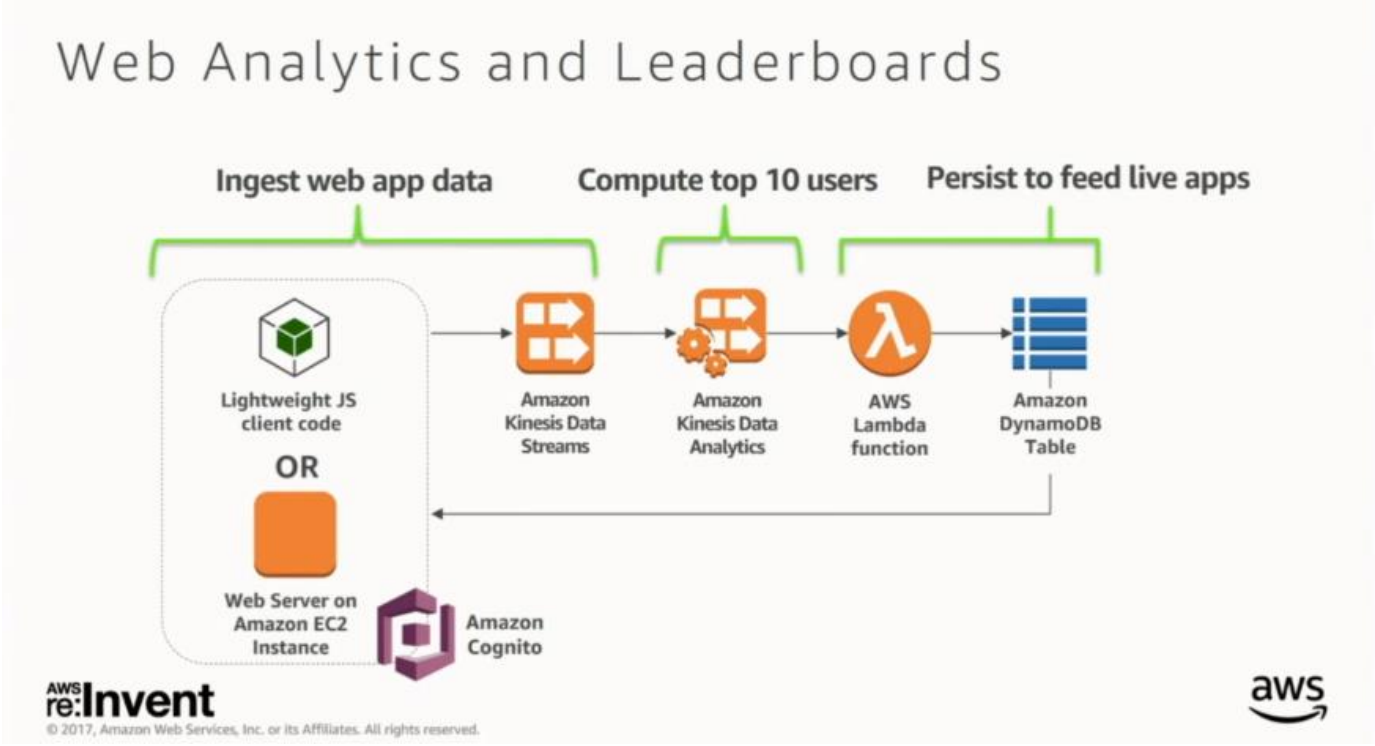
Easily write SQL code to process streaming data

Continuously deliver SQL results

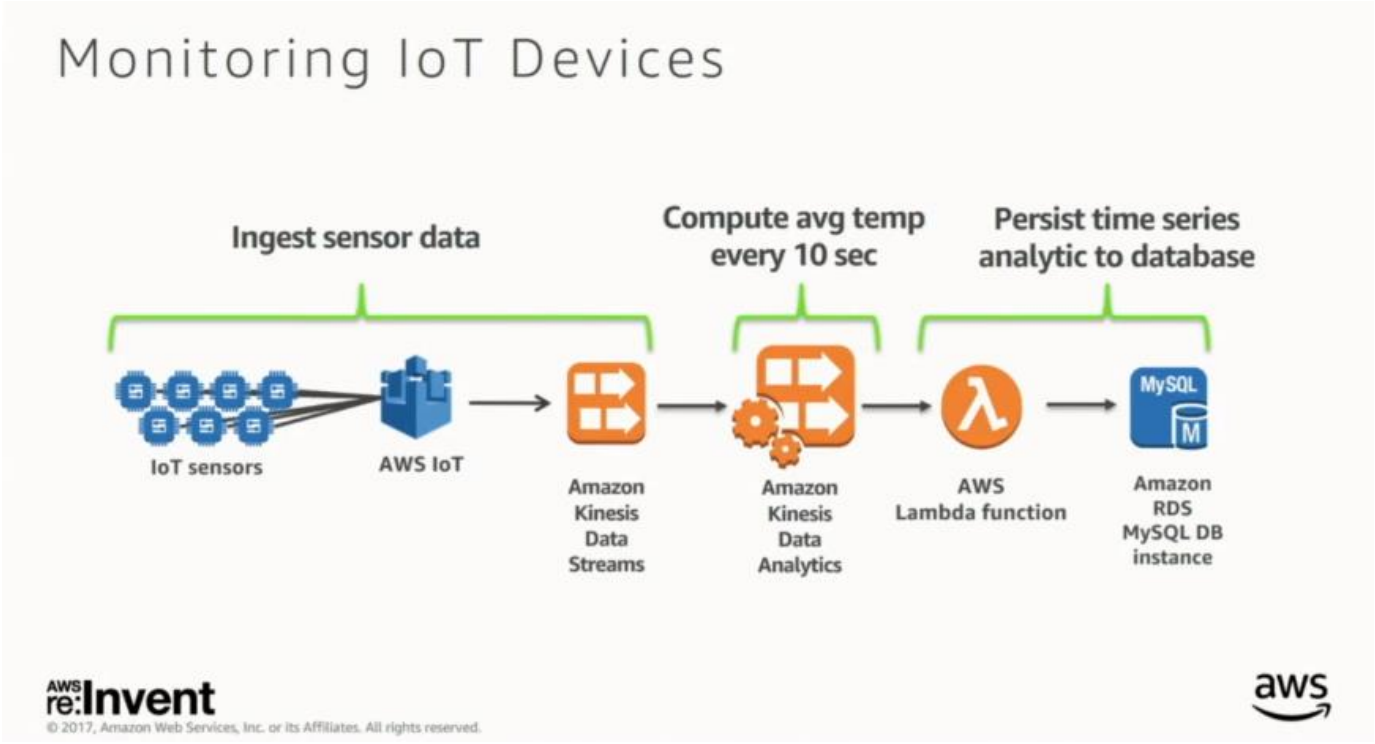
Common Use Cases

Three Common Scenarios

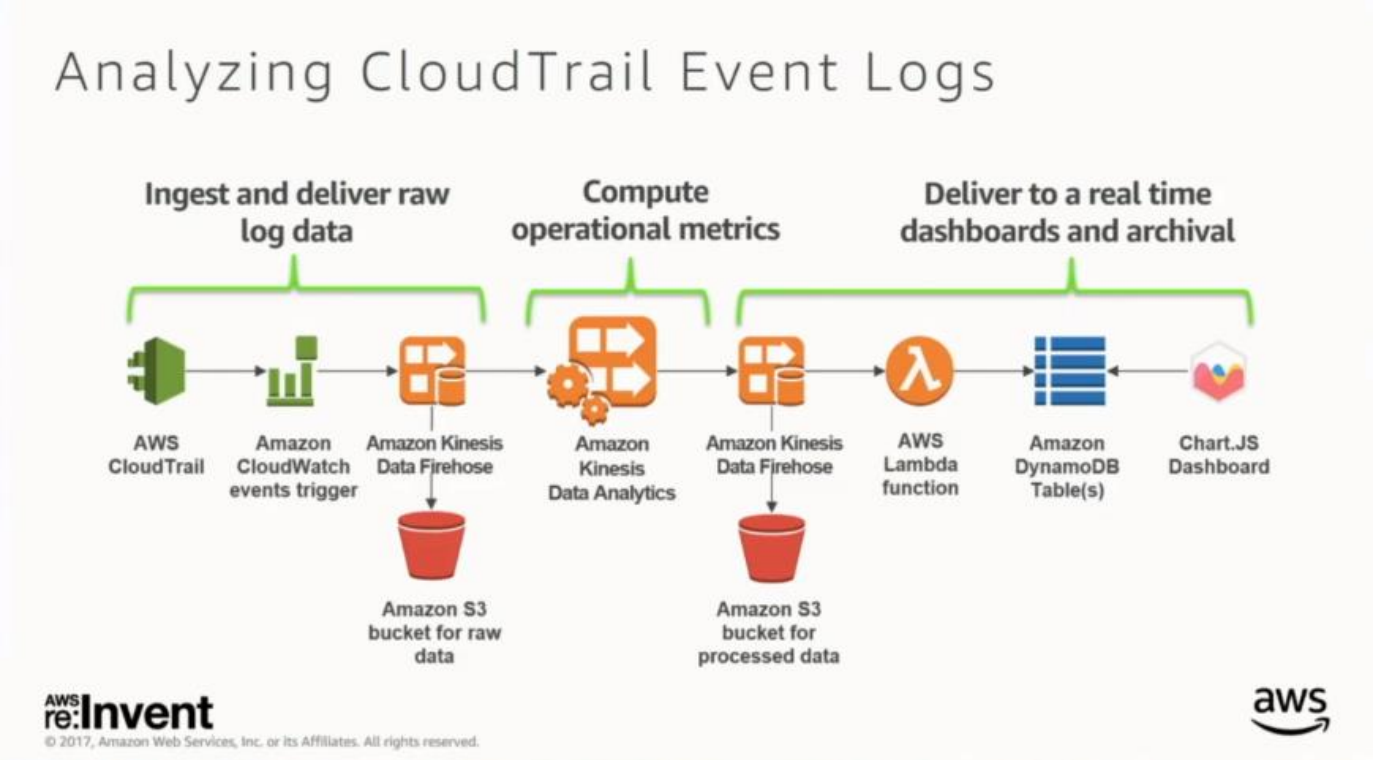
Streaming Ingest-Transform-Load	Deliver data to analytics tools faster and cheaper
Continuous Metric Generation	Compute analytics as the data is generated
Actionable Insights	React to analytics based off of insights



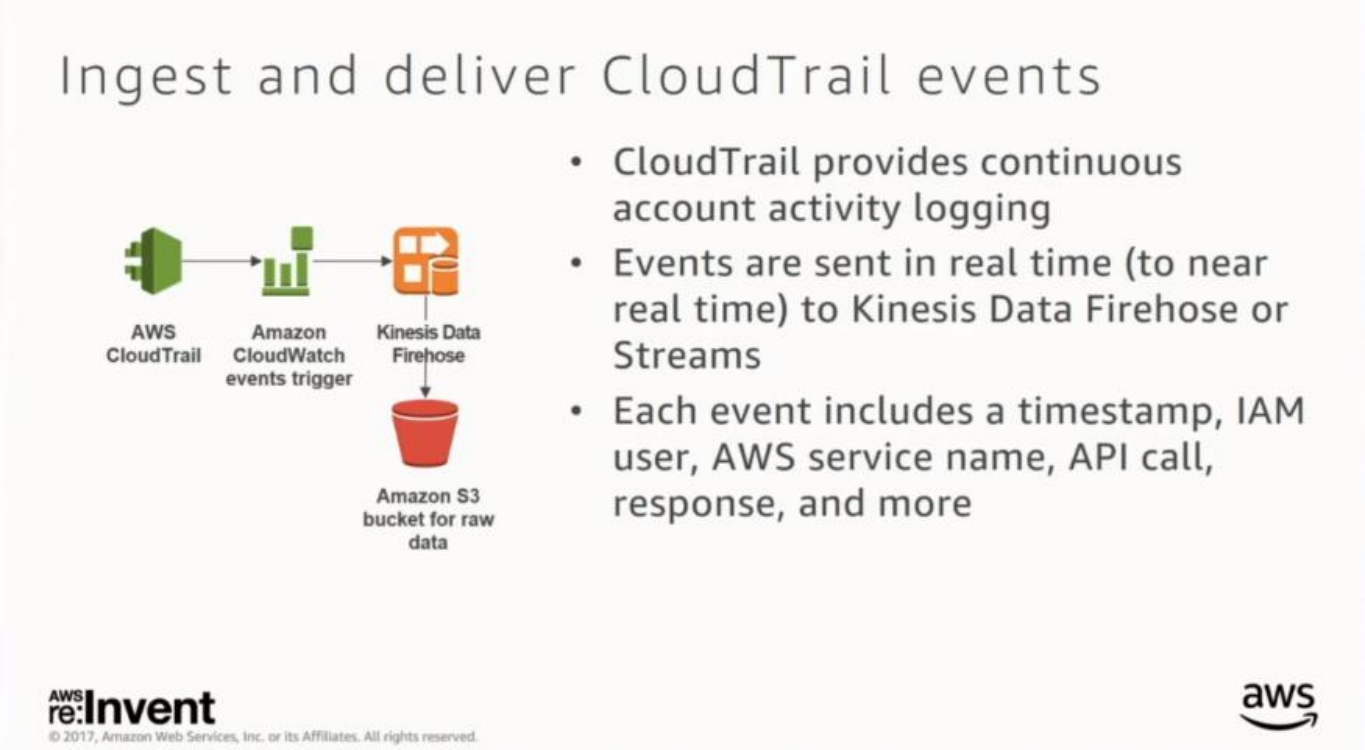
When you write data to a Kinesis Stream, there are 3 common parameters you need. The stream name or where you are going to store the events in the temporary buffer that Kinesis data streams provide, the piece of data that you are sending like an event or a group of micro-batched events to be written to the kinesis data stream, the last thing is a partition key which is a key associated with the data that you supply and we can use to determine where the piece of data is logically stored in the Kinesis stream.



You can configure a very simple Rule in AWS IoT to forward data to a Kinesis Data Stream, you can forward a group of data to different Kinesis Data Streams using different rules.

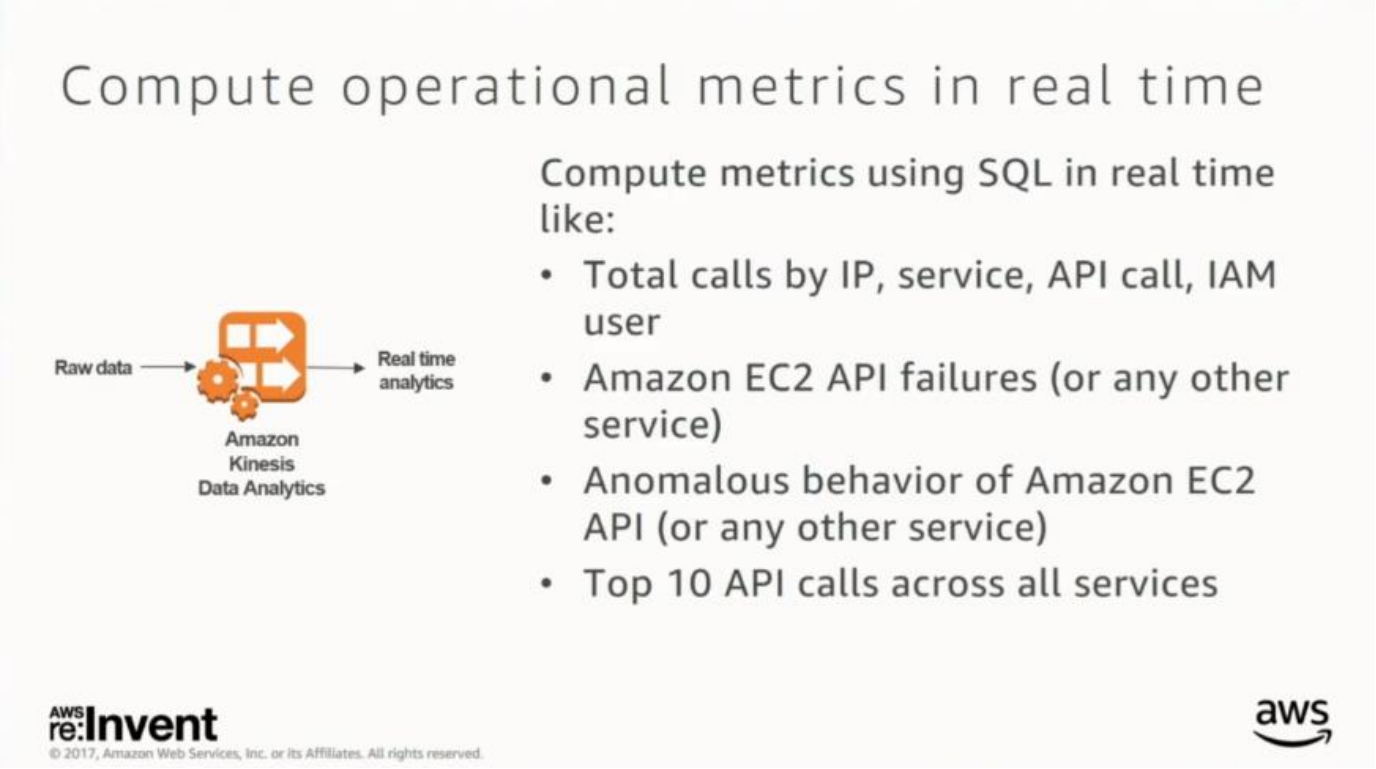


Kinesis Data Streams has integrations with some AWS services like AWS CloudTrail where we can set up an Amazon CloudWatch event trigger to take data that is generated by AWS CloudTrail and send it to Amazon Kinesis Data Firehose. AWS CloudTrail is a service that logs activities that occur within your AWS account, it can log the API calls you make with the SDK as well as the associated metadata like the IAM user making the calls, log data, etc.

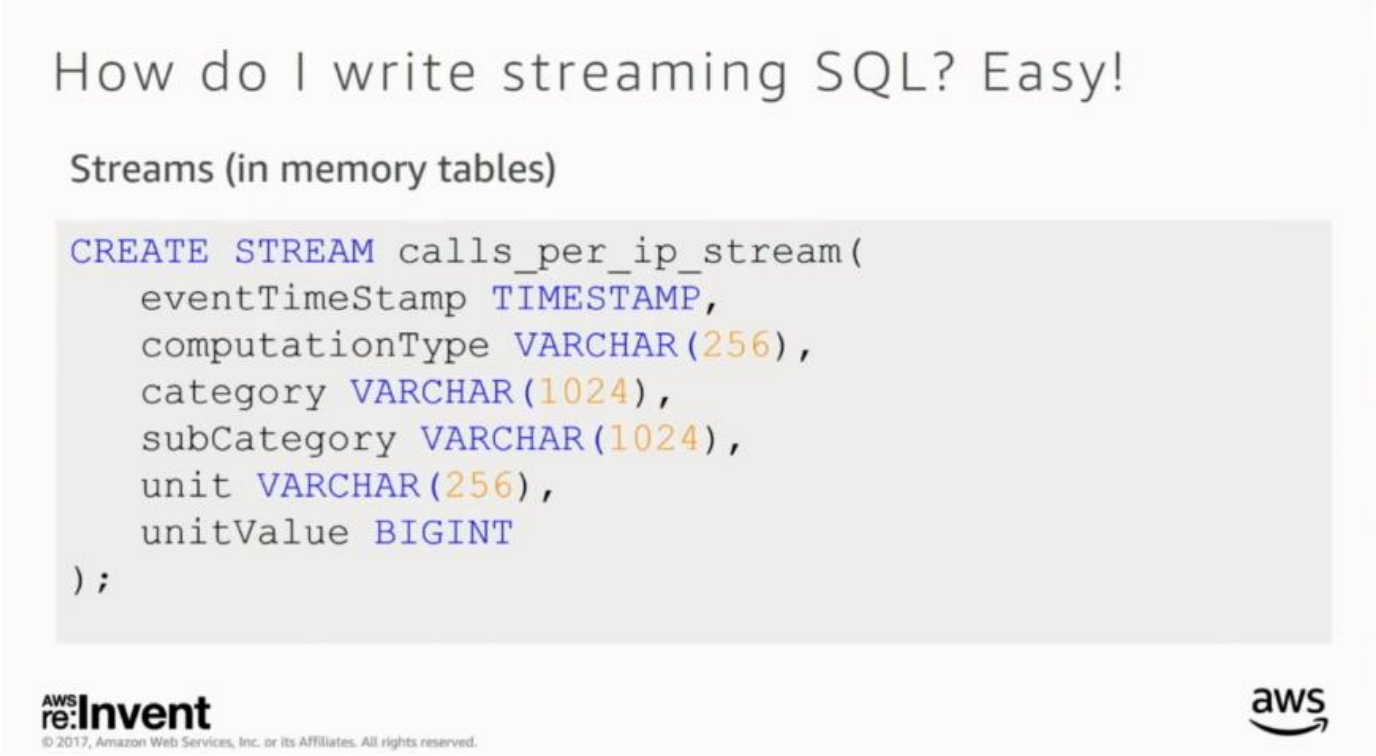


We may set our ***buffering hints*** for the Firehose to be like 1 hour or a very large file size so that we get it to send batched events data to the raw S3 bucket in large chunks.

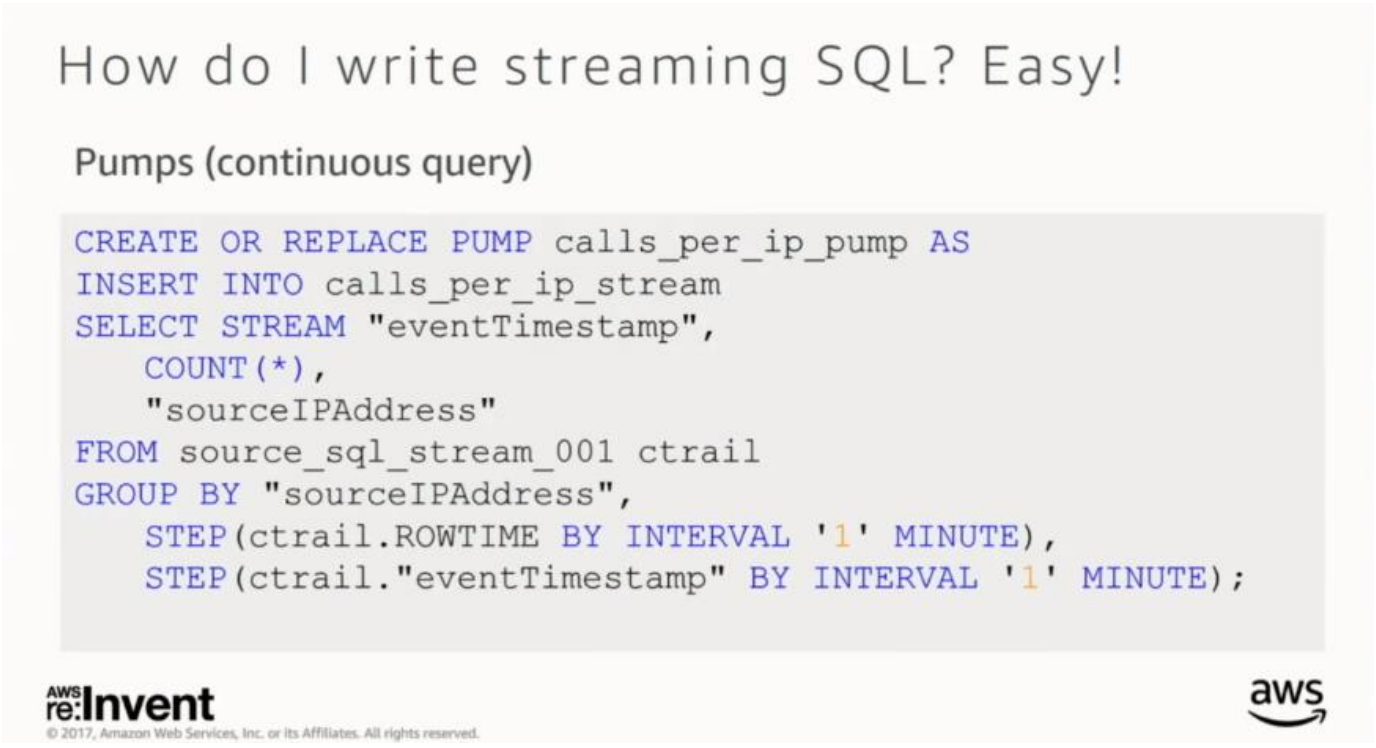
There are several ways to get data into Kinesis, a very common pattern is using API Gateway to forward data to a Kinesis Streams.



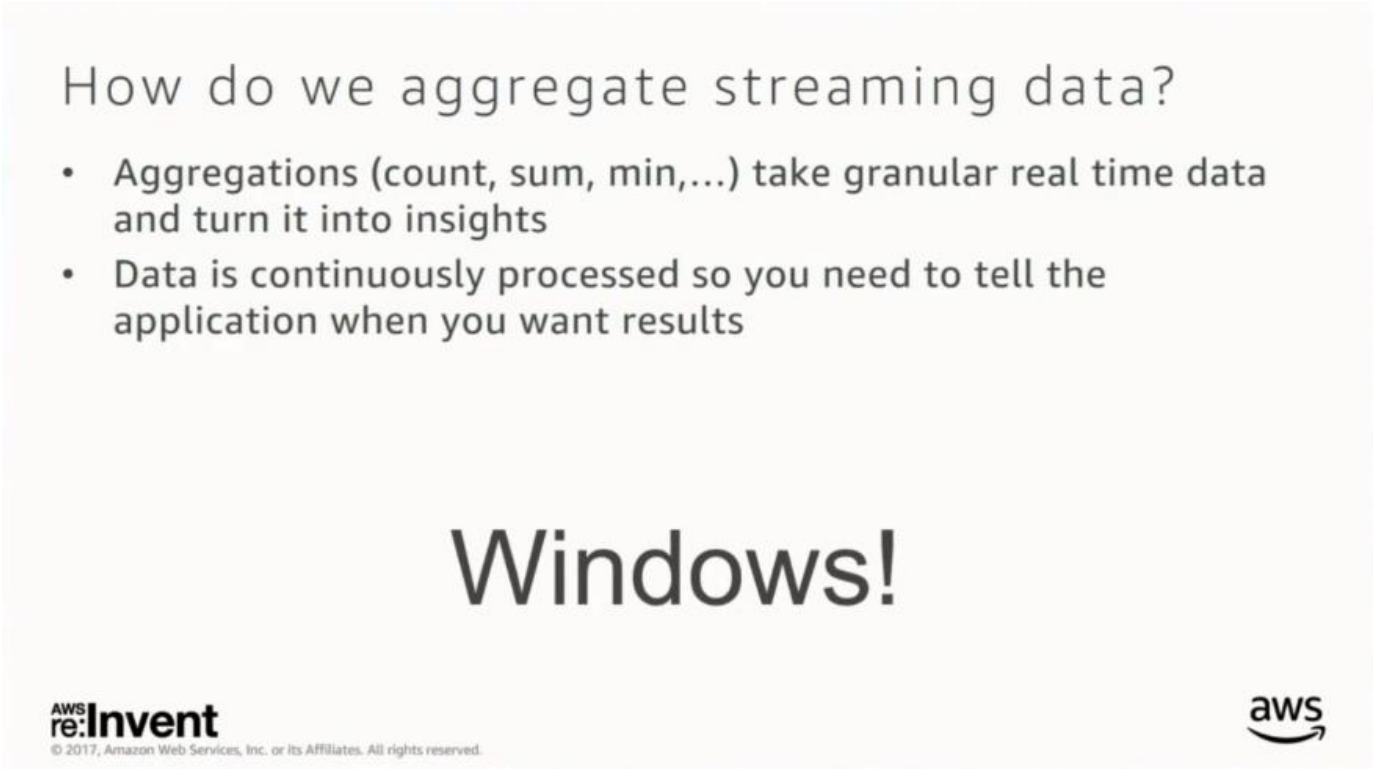
After ingesting the data into the Kinesis Streams, we now compute some operational metrics on that real-time data. we are going to see how to process the data using SQL code.



There are 3 main concepts when you build a Kinesis Data Analytics application, they are the Source, the Code, and the Destination. AWS will continuously read data from the source, continuously process that data with your code, and continuously emit the results to the destination for you. There are 2 components to the code, there are in-memory objects called in-application streams that are like materialized view on the stream. This detects JSON data and automatically map them to a stream having column types for the data value types.

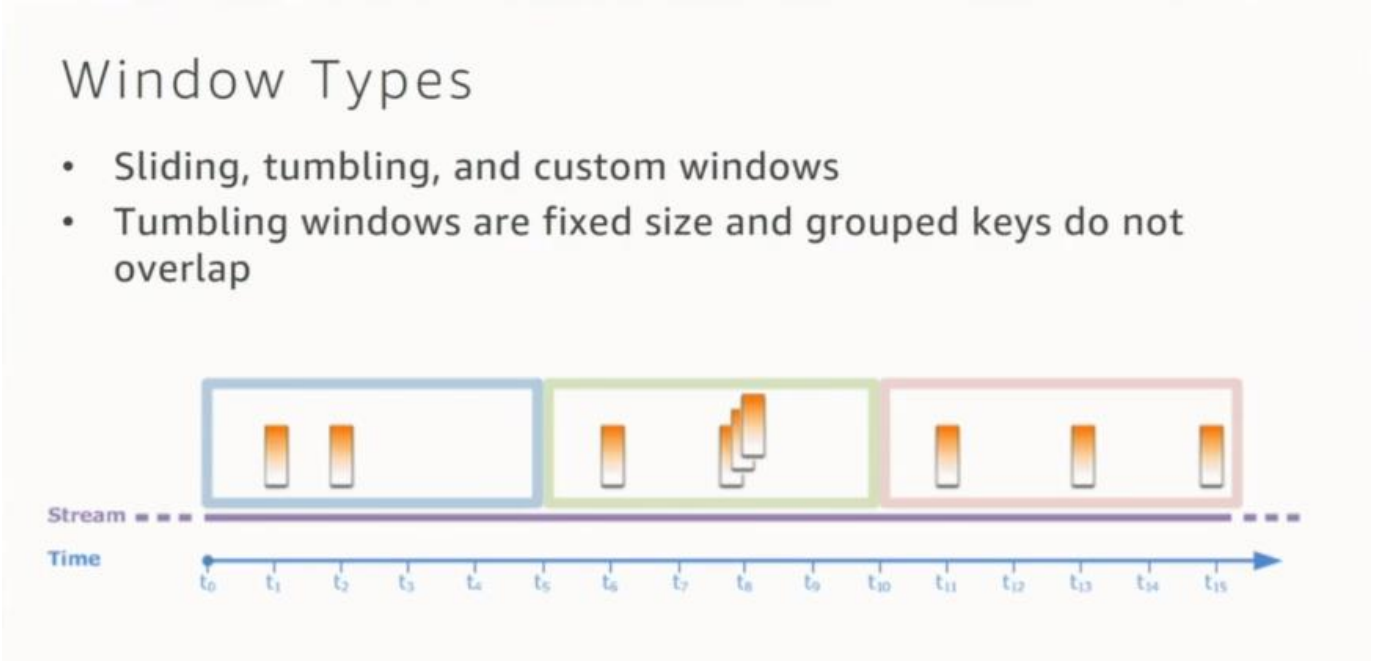


To get it to actually process the data we use the 2nd concept called the Pump, this is basically a continuous SELECT statements that are inserted into the stream. The first step is mostly a De-dupe step, then we might do some analytics over the data. we are grouping by a key like the sourceIPAddress, then 2 step functions ask for results every 1 minute and have the results grouped by the timestamp.



The step functions are called windows, when you do a count, sum, min, max, these are called windowed operators. What we saw earlier was tumbling windows, but there are also sliding windows, session windows, hopping windows, and other custom windows. A window defines the bounds where we are bounding a result set by time, key. A window defines where to start and stop your computation in a continuous data stream.

In the batch context, the window is the entire data batch size you are working on. In the streaming context, the window is what you specify when doing things like aggregations.



A ***tumbling window*** is a fixed size (here defined by time) and grouped keys do not overlap, meaning that the count results will be 2 (at time =t5), 4 (at time = t10), 3 (at time t15). On the other hand, a sliding window is overlapping and computes a rolling window with overlapping times. A session window is overlapping and not fixed sized, it is like a user session on your website.

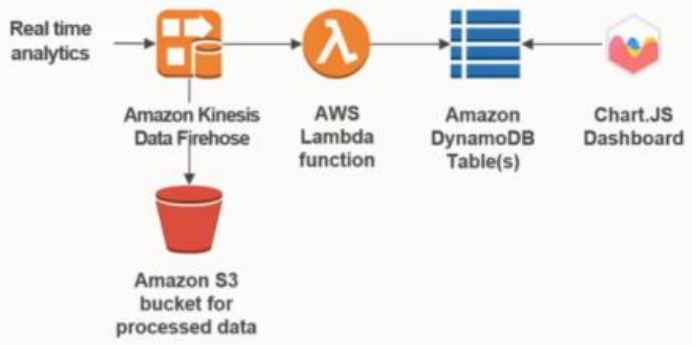
Event, ingest, and processing time

- Event time is the timestamp is assigned when the event occurred, also called client-side time.
- Processing time is when your application reads and analyzes the data (ROWTIME).

```
...
GROUP BY "sourceIPAddress",
    /* Trigger for results */
    STEP(ctrail.ROWTIME BY INTERVAL '1' MINUTE),
    /* A timestamp grouping key */
    STEP(ctrail."eventTimestamp" BY INTERVAL '1' MINUTE);
```


With the code above, you will get a processed result every 1 minute.

Persist data for real time dashboards



- Use Kinesis Data Firehose to archive processed to in S3
- Use AWS Lambda to deliver data to DynamoDB (or another database)
- Open source or other tools to visualize the data

We are doing ***long term data archival of the processed data in S3***. We then use a lambda function to update the DynamoDB table(s).

Late results

- An event is late if it arrives after the computation for which it logically belongs to has been completed
- Your Kinesis Analytics application will produce an amendment

```
…
GROUP BY "sourceIPAddress",
    /* Trigger for results */
    STEP(ctrail.ROWTIME BY INTERVAL '1' MINUTE),
    /* A timestamp grouping key */
    STEP(ctrail."eventTimestamp" BY INTERVAL '1' MINUTE);
```


Updating a database

- Perform inserts but on duplicate key update
- For DyanamoDB, here is the AWS Lambda code:

```
…
GROUP BY "sourceIPAddress",
    /* Trigger for results */
    STEP(ctrail.ROWTIME BY INTERVAL '1' MINUTE),
    /* A timestamp grouping key */
    STEP(ctrail."eventTimestamp" BY INTERVAL '1' MINUTE);
```

What does all this cost?

- All services used in the solution are pay as you go
- All services used are serverless and have lower devops expense
- This solution will cost the “average” customer less than:

$100 per month

Where do go next?

Try it out yourself

Go to aws.amazon.com/kinesis/

Some good examples:

- Get started in minutes with a clickthrough template for AWS CloudTrail Event Log Analytics - <link> (friendly URL)
- Tinyurl.com/rt-dashboard
- Great blog posts with example use cases

Lots of customer examples

Amazon Kinesis as Databus - Migrate from Kafka to Kinesis| Enterprise

1 billion events/wk from connected devices | IoT

17 PB of game data per season | Entertainment

80 billion ad impressions/day, 30 ms response time | Ad Tech

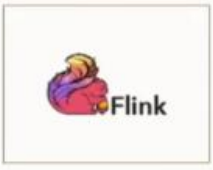
100 GB/day click streams from 250+ sites | Enterprise

50 billion ad impressions/day sub-50 ms responses | Ad Tech

10 million events/day | Retail

Funnel all production events through Amazon Kinesis

Integrate with your current solutions



Get help from partner systems integrators

slalom

47 Lining

8K Miles

apps associates
extreme expertise

clearscale

NorthBay

BEEVA

class
method