Building a Serverless Stream Analytics Platform with Amazon Kinesis Data Firehose and MongoDB Realm

by Vasanth Kumar | on 24 NOV 2021 | in Advanced (300), Analytics, Customer Solutions, Kinesis Data Analytics, Kinesis Data Firehose, Serverless, Technical How-To | Permalink | Comments | Share

By Vasanth Kumar, Principal Solutions Architect – MongoDB

Most enterprise companies are building a serverless architecture or migrating their current IT solutions to one.

This allows developers to concentrate on core product behavior rather than spending excessive time managing infrastructure or deploying and operating underlying tech stacks.

A serverless architecture strategy reduces complexity and provides more flexibility in adopting the features and non-functional requirements needed to support market agility.

In this post, I will look at an example of an Internet of Things (IoT) use case and build a serverless scalable platform using Amazon Kinesis Data Firehose, Amazon Kinesis Data Analytics, and MongoDB Realm. You'll learn how easy it is to develop mobile and desktop applications on top of the data platform for different personas.

Before getting into the solution and reference architecture, let's review each component.

Why MongoDB Realm?

The Developer Challenge

Developers typically must build secure application servers that implement features like user authentication, data validation, and business logic on top of the database. These servers always need an API so they can communicate with browsers and mobile applications or middleware components.

Each of these problems can be solved in isolation with a wide variety of libraries, frameworks, and services—but the challenge of deciding which solution solves each problem, with the right tradeoffs for your use case, can be daunting.

MongoDB Realm for Backend Developers

MongoDB Realm is a server-side application that doesn't require developers to set up and manage server infrastructure, including provisioning, deployment, operating systems, web servers, logging, backups, redundancy and security.

This allows developers to shift their focus from implementing boilerplate backend components to writing the code that makes an app unique.

MongoDB Realm for Mobile Developers

MongoDB Realm helps mobile development by handling unpredictable environments in which connections can be lost and devices shut down unexpectedly.

Realm seamlessly integrates your mobile client, backend APIs, and Atlas Cluster (fully managed database as a service). It takes care of security vulnerabilities across all components in an ecosystem, and consistently serializes objects between networks, database storage, and application memory.

Realm Database solves many common mobile and edge headaches, including:

- **Local storage:** Realm Database runs directly on client devices, and objects can be accessed using the native query language for each platform.
- **Network reliability:** Realm Database is offline-first. You always read from and write to the local database, not over the network. When Realm Sync is enabled, Realm Database synchronizes data with MongoDB Realm over the network in a background thread, pushing local data changes to MongoDB Realm and pulling down remote changes.

Why Amazon Kinesis?

Amazon Kinesis is a fully managed service for real-time processing of streaming data at any scale. It provides a serverless platform that easily collects, processes, and analyzes data in real-time so you can get timely insights and react quickly to new information.

With Amazon Kinesis, you can ingest data such as video, audio, application logs, website clickstreams, and IoT telemetry data into databases and object stores. Kinesis can handle any amount of streaming data and process data from hundreds of thousands of sources with low latencies.

Amazon Kinesis Data Firehose provides the easiest way to ingest, transform, and load (ITL) data streams into AWS data stores (such as Amazon S3, Amazon Redshift, ElasticSearch, or Splunk) for near real-time analytics with existing business intelligence tools.

Kinesis Data Firehose also has built-in support for no-code data transformation functions before loading your data in the right destination tool. For advanced data transformation, Kinesis Data Firehose integrates with AWS Lambda and Amazon Kinesis Data Analytics.

Kinesis Data Analytics delivers the easiest way to process data streams in real-time with Spark jobs or Apache Flink without having to learn new programming languages or processing frameworks.

Real-World Use Case: Building Intelligent Tolling Systems

The Challenge

Every day, millions of commuters across the United States drive through toll booths. Consider the fictitious "AnyCompany," whose main business is facilitating this daily process by building intelligent tolling systems (ITS) like the ones outlined by the U.S. Department of Transportation.

The primary goal of an intelligent tolling system is to collect electronic tolls while also increasing safety, improving ROI, and creating a seamless experience via mobile payments, touchless entry and exit, free-flow traffic management, and so on.

To achieve this, AnyCompany must be able to automate processes, as well as track several metrics in real-time—from the

Resources

Why Work with AWS Partners
Training for Partners
AWS Competency Partners
Managed Service Providers (MSPs)
Partner Central Login
Case Studies and References
AWS Sponsorship Opportunities

Follow

AWS Partners
AWS Cloud
APN LinkedIn
APN YouTube
RSS Feed
APN Email Updates

AWS Partner Paths
A curated journey through AWS Partner resources, benefits, and programs.
Learn more »

number of vehicles at a given time passing through a tolling booth, to traffic density in the road, types of vehicles, and number of lanes occupied. But that's not all.

They'll also need to implement the security layer, diagnostics, and take care of other non-functional requirements like scaling and high availability. On top of it, managing, supporting, and keeping the features in line with the market is a massive task in and of itself.

Building an ITS platform in-house requires:

- Dedicated development and IT team.
- Substantial upfront investments.
- Vast skill sets across various computing and database technologies to centrally manage huge amounts of data coming from distributed systems.
- Time to roll out the platform, typically 9-18 months.

So, how should AnyCompany get started?

The Solution

By using Amazon Kinesis Data Firehose, Amazon Kinesis Data Analytics, and Realm with Atlas, AnyCompany can quickly build and roll out a low-code ITS platform.

This solution helps them by lowering development cost, speeding time to market, and reducing operational costs. It also provides high safety and security to the ITS owners and end customers. Fully automated deployment of these components via seamless DevOps models, ensures agile development cycles.

The core functions required to build the platform are:

- Streaming engine to stream data from an edge device located at the toll booth.
- Transformation engine which can transform the raw data to the interpretable and processable format.
- Analytical engine for real-time analytics like detecting black-listed vehicles.
- Data source for capturing huge sets of high frequency telemetry and event data and analytical insights.
- REST APIs to expose the data as a service, which will be consumed by multiple channels including mobile applications.

The architecture diagram below explains how to build a centralized toll management platform for AnyCompany.

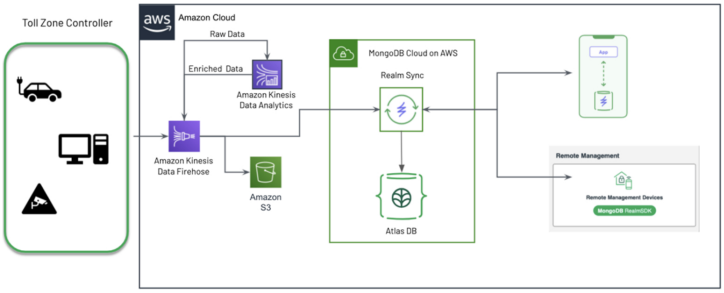

Figure 1 – Centralized toll management platform.

The toll zone controller at the toll booth is configured to send the telemetric and event data (tolling transactions) to Amazon Kinesis Data Firehose, which automatically delivers it to a configured destination.

Kinesis Data Firehose can also be configured to transform the data before data delivery. AWS offers software development kits (SDKs) for many popular programming languages, each of which provides an API for Kinesis Data Firehose.

Realm Platform is a cloud-hosted backend-as-a-service (BaaS) that provides a serverless application platform which takes care of the details of deployment and scaling. Realm Sync shares data changes between a Realm-linked Atlas cluster and AnyCompany's mobile applications.

Let's walk through each integration point in the architecture above.

A Developer's Guide to How it Works

Data Ingestion for the Telemetric and Event Data

To get started, first create a data ingestion API with Realm and configure this API as the destination for the data delivery from Kinesis Data Firehose. Refer to this AWS blog post for detailed steps on how to create Realm API and integrate with Amazon Kinesis Data Firehose.

Next, configure an Amazon S3 bucket as another destination to hold all of the images.

Sample payload coming from edge controller contains vehicle data passing in the toll:

```
{
  "transactions": {
"id": "12345678901234567890123456789012345678", "transactionDate":   "2020-12-01T21:25:43.51
    "laneNumber": "1", "laneSequence": "8765", …..
    },
    "vehicle": {
      "vehicleSpeedOls": 73 , …....
    },
    "images": [
      {
        "cameraType": "REAR","cameraNumber": 1, "contentType": "JPEG",
        "uri": "https://s3.amazonaws.com/its/Veh_12334.jpg ",
        "transactionPlate": {
          "state": "AZ", "number": "ABC123", "numberConfidence": 90
        }
      } ]
  ……..
    }
```

Sample script for Realm API data ingestion:

```
  if (firehoseAccessKey == context.values.get("IOTPOC_SECRET_KEY")) {
```

```
        var collection = context.services.get("mongodb-atlas").db("mongoiotpoc").collection(
        fullDocument.records.forEach((record) => {
              const document = JSON.parse(decodeBase64(record.data))
              document.transaction.transactionDate = new Date(document.transaction.transacti
              document.date = new Date(document.date);
              const status = collection.updateOne({"date": document.date, "tollPointId":  dc
        })
```

Please refer to this GitHub repository for Realm app code and sample payload.

Amazon Kinesis Data Analytics Integration with Kinesis Data Firehose

Event data from the edge controller is passed to Amazon Kinesis Data Analytics to conduct real-time and complex data analytics like vehicle theft detection, revenue patterns over period, traffic prediction and revenue forecast.

Users can interactively query streaming data using standard SQL, build Apache Flink applications using Java/Python/Scala, and build Apache Beam applications using Java to analyze data streams.

In this scenario, we have built simple analytics (Windows operation) and enriched the data with geospatial coordinates using Flink. Source for the analytics is "**tolldata-stream**" (Kinesis Data Firehose service) and sink is another Kinesis Data Firehose service "**processed-IOT-stream**".

A Flink application can be created using preferred programming language like Scala or Java and create a Kinesis consumer and producer.

In the main method: consume the input stream, process the data and add it to the producer.

Sample code:

```
DataStream<String> input =
env.addSource(new FlinkKinesisConsumer<>(inputStreamName, new SimpleStringSchema(), inputF

FlinkKinesisFirehoseProducer<String> sink =
new  FlinkKinesisFirehoseProducer<>(outputDeliveryStreamName,
        new SimpleStringSchema(), outputProperties);
              input.map(value -> {
        JsonNode nodeDetails = jsonParser.readValue(value, JsonNode.class);
        return new Tuple2<>(nodeDetails.get("tollPointId").asText(), 1);
          }).returns(Types.TUPLE(Types.STRING, Types.INT)).keyBy(value ->
                            value.f0).timeWindow(Time.seconds(60)).sum(1)
                .map(value -> "---  Add derivation code  ------")
                .addSink(createFirehoseSinkFromStaticConfig());
```

The image below illustrates the configuration of the above mentioned Flink application as an analytics job.

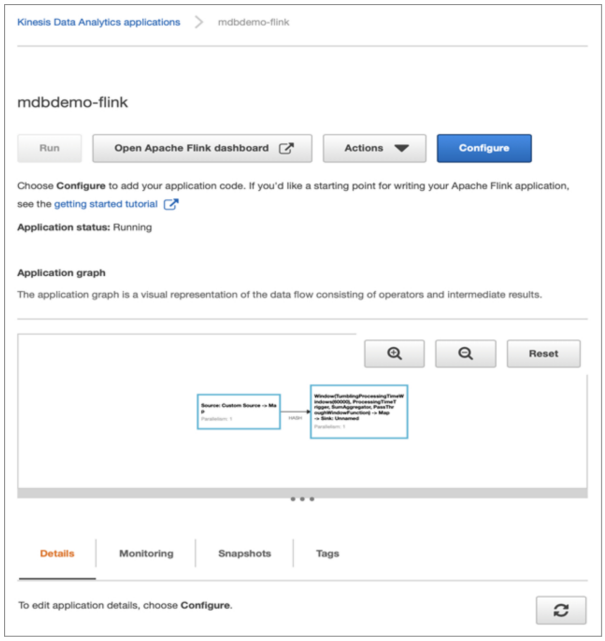

Figure 2 – Analytics using Flink.

Data Ingestion for Insights Derived from Amazon Kinesis Data Analytics

This configuration is similar to the steps mentioned for the "Data Ingestion for the Telemetric and Event Data" where we'll be configuring "**processed-iotstream**" with a different Realm data ingestion API to capture the analyzed insights into MongoDB Atlas cluster.

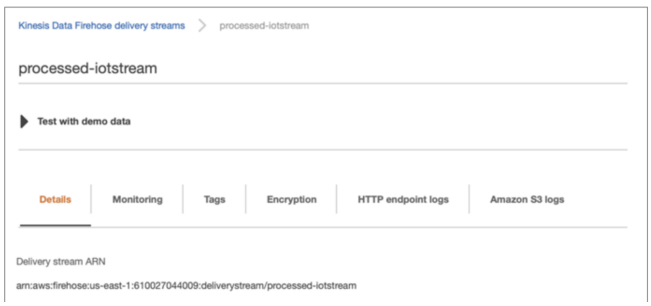

Figure 3 – Configuring processed data to persist into Atlas.

APIs for the Mobile and Desktop Applications as Per the Features

With all of the data in place with MongoDB Atlas, customers can build Realm serverless backend APIs based on the business use cases. Please find the detailed steps in this AWS blog post.

Now, what's left is to build the user interface (UI) artifacts and integrate with Realm Database and Sync APIs to deliver mobile and web applications. Also, you can integrate the charts and dashboards built on top of the data just by configuring different dimensions.

Below is the sample chart built on toll data with almost no effort. To create the chart, simply select the Geo Information System (GIS) chart type in Atlas Charts. Then, configure window function (sum of vehicles, for example) against the field toll ID.

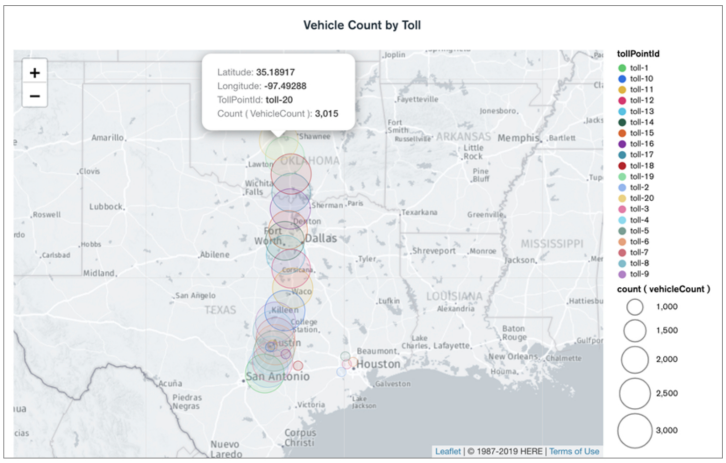

Figure 4 – Sample dashboard built using MongoDB charts on top of processed data from analytics.

Conclusion

To summarize, we have built an enterprise cloud-native serverless platform where data is streamed, analyzed, enriched, persisted, visualized, and exposed as a service to build web and mobile applications for multiple personas.

As part of this post, I have taken the example of an IoT but this solution can be used for any use case which involves data streaming, processing, running analytics, and exposing data as a service to build digital channels for customers.

Contact the MongoDB partners team and speak with experts to learn more about building serverless platforms for data streaming, processing, and data management use cases.