Born to Perform: build, race, analyze, repeat.

21 October 2022 - 8 min. read

AWS IoT Core | AWS Lambda | Data Ingestion | Data Lake | Database | OpenSearch | Serverless

In the last few years, **data-centric projects** have been one of our primary focuses and interests.

Due to companies' growing curiosity around this topic and higher demand, we strongly committed to **R&D activities to test our data expertise** to the limit.

For us, understanding things it's not enough. We need to experiment, fine-tune and get our hands dirty in bringing tech into extreme and crazy contexts.

And that's how, some time ago, we end up assembling a hand-made realistic racing simulator. We carefully equipped it with a direct-drive steering wheel with 32 Nm of torque, pedals with hydraulic dampers designed to withstand over 140 kg of pressure, a racing seat with 6-point belts, and, above all, a 4DOF movement platform to reproduce vehicle dynamics and roughness of the track.

As a next step, we built **super-powerful hardware able to keep up with all our performance obsessions**: we installed "Assetto Corsa Competizione" on it, one of the most accurate driving simulation software ever built.

The most exciting part of the project - and the most challenging one - however, was **intercepting as many metrics as possible from "Assetto Corsa" and bringing them efficiently to the Cloud through a high-performing data ingestion system**. Last but

not least, we aimed to build a **detailed dashboard reporting data in real-time** during the lap so that the driver could adapt the driving and improve.

Cool, right?

Galileo Galilei stated: "Measure what is measurable, and make measurable what is not".

You know, we only get inspired by the best;)

This Cloud-side of the project will be the focus of the following paragraphs. We'll describe in detail how we chose and combined AWS services for data-centric projects to succeed.

Lights out and away we go!

To detail how we developed the data analytics solution for our Simulator Project,

we will present why we have chosen specific AWS Services and how we have combined them to:

- create the infrastructure for extracting data from the Simulator and storing it in the Cloud
- run the ETL process
- analyze the information and extract value from the data

Technical challenges

Since the beginning, we only had one clear objective in mind: maximizing performance. We needed a fast and simple way to extract all the possible metrics from Assetto Corse Competizione.

We immediately faced our first challenge: the racing simulator was born just for playing purposes, so it couldn't integrate with any third parties applications by default. After hours and hours spent reading an endless number of documentation pages and forums, we finally discovered that this software natively exposes a UDP local server. As we needed to extract runtime data super-efficiently, the UDP protocol was perfect for us.

Bingo! That was easy: we just had to create a Python connector. BUT... Unfortunately, once we analyzed the first batch of data, we noticed that many metrics were missing. So we needed to find another way.

After some research, we opted to read the information we needed directly from the RAM. First of all, we developed a lean Python script to extract and convert the binary data and verify that we could access everything we needed.

Eureka! This time nothing was missing.

So we added the possibility of packing the extracted data into JSON and pushing them directly to the Cloud to our script.

Ok, let's move them to the Cloud! In the Cloud, but... where?

The AWS Architecture

Ingestion

If you are a regular reader of this blog, you know that in beSharp we love to design high-performant, scalable, and resilient Cloud infrastructures. And this project seemed to be cut out for us here.

Which AWS service would be the right one to ingest data?

We had multiple options: we could manage the whole data flow coming from the Simulator with a Kinesis Stream or with a Fargate cluster.

In both cases, scalability, and performance would have been achieved, but the need of sorting input traffic through software developed and managed directly by us was still open.

Since we wanted a completely event-driven and serverless infrastructure, these solutions mismatched our quality criteria.

We need to ensure a high-scalability and "out-of-the-box" integrations with several AWS Services.

We opted then for **AWS IoT Core**. It ensured the following:

- the possibility of implementing fully-manged rules to sort traffic based on the content
- compatibility with real-time data analysis services

Once understood how to bring data into our AWS infrastructure, we needed to split the huge stream of raw data into 3 parts:

1. Ingestion system for real-time analysis;
2. lap time-related data ingestion;
3. Business analysis system.

Ingestion system for real-time analysis: necessary for real-time metrics visualization. We aimed to develop a service allowing near real-time data visualization of all the metrics coming from a custom AWS IoT Core rule with no need for infrastructural management effort.

There are two main AWS services right for this use case: Amazon QuickSight e OpenSearch.

Amazon QuickSight allows you to create dashboards for advanced reporting with many cool graphic objects. The downside is that, unfortunately, it cannot integrate with AWS IoT Core unless you pass the data coming from the AWS IoT Core rule to a Kinesis Stream and then put it in a data lake on Amazon S3. Since Amazon QuickSight is mainly a service used for Business Analytics purposes with no need for Real-Time Analysis, we considered using Amazon OpenSearch instead.

Amazon OpenSearch gives the same possibilities as Amazon Quicksight in terms of graphic objects available for advanced reporting, but, what makes this service a game-changer for our case, is its native integration with our rule on IoT Core. To connect Amazon OpenSearch to AWS IoT Core we only needed to set our Amazon OpenSearch cluster and the data destination index as a target for our rule.

The only thing we have to do is to set as the target of the rule both the endpoint of our Amazon OpenSearch cluster and the destination index.
We have a winner!

Lap time-related data ingestion: forwarding data gathered from the track thanks to AWS IoT Core rules has been simple. This rule selects in-transit data of every single lap and then invokes a computational component able to bring processed data into storage service.

For this stream, we need two different services: a computational component and a storage service.

Computational Component

AWS provides many options to develop and execute software directly in Cloud, but it is important to keep in mind our purpose of getting a fully serverless, highly scalable infrastructure benefitting from a Pay-as-you-go pricing model which is natively integrable with AWS IoT core rules.

That said, Fargate and Lambda are the only options meeting our needs.

On one hand, Fargate would have allowed us to create a Docker's containers cluster, but we would have to manually set up and manage the scalability. Furthermore, a Fargate cluster wouldn't switch off automatically when not invoked. This would have brought a considerable impact on the spending.

AWS Lambda Function on the other hand is fully compliant with our pay-as-you-use requirement, as it switches off once the runtime has terminated and it turns on only with new invocations.

Database

We needed an easily queryable, highly performant, fully-managed, and scalable storage to be used in a pay-as-you-use model.

The two viable options are Amazon Aurora Serverless and DynamoDB.

Amazon Aurora Serverless would have brought us to a cluster with an always online node with an impact on the billing. Furthermore in this case there is no need for the management of structured data with relations.

For this reason, we opted for a single-table Amazon DynamoDB. This provided a key-value NoSQL database allowing data memorizations for time-laps and leaderboards.

Data Lake

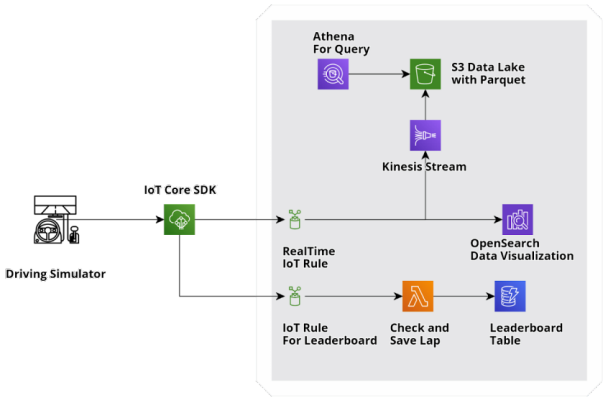
Metrics ingestion for historical purposes, and future analysis (and Machine Learning): this third stream is about the data lake. Our solution generates a huge amount of data that must be available for future analysis and manipulation, without the need of modifying the infrastructure.

So, which is the best place to store the data? We needed a scalable, pay-as-you-use storage service, with different security levels possible and different lifecycle policies.

Amazon S3 was the best choice to meet our needs.

The last thing we needed to do was to write data directly on the S3 bucket, without the need of reading from AWS IoT Core or using a custom application to write on Amazon S3. This would have helped us avoid data transfer management.

The resulting infrastructure for our data-driven project is displayed in the picture below. It allows any future business analysis using services like Amazon Athena or Amazon Quicksight, both natively integrated with Amazon S3.



Wrap-up

You may wonder, "did you keep this project to yourself?"

Of course, not :)

Since the very beginning, we have already had big things in mind. It was only a matter of defining the road (well, the track) to go.

After a few months, we were finally at the **AWS Summit 2022 in Milan**, the Italian Cloud event by definition and a huge chance for us to test our outcome. We can proudly say that the Sim was the undisputed star of the whole event.

It is not a simple starting point. And it is not a finish line. Instead, it is part of a much broader ambitious project called "**beSharp Performance Technology**", "bPT" for friends.

bPT is the newborn **R&D-specific brand by beSharp** that will carry out our approach to making things work in extreme contexts.

For the curious one, here is what's behind bPT's scenes.

beSharp declines all responsibility for any nerditude attack after reading this article ... but it is very well-disposed to know in the comments what you think of all this crazy project!

See you soon on **Proud2beCloud**!

About Proud2beCloud

Proud2beCloud is a blog by beSharp, an Italian APN Premier Consulting Partner expert in designing, implementing, and managing complex Cloud infrastructures and advanced services on AWS. Before being writers, we are Cloud Experts working daily with AWS services since 2007. We are hungry readers, innovative builders, and gem-seekers. On Proud2beCloud, we regularly share our best AWS pro tips, configuration insights, in-depth news, tips&tricks, how-tos, and many other resources. Take part in the discussion!

Paolo Di Ciaula

DevOps Engineer, Frontend Developer, and Mobile App Developer @ beSharp.I divide my free time between good music, development, and... beer (sometimes mixed for better results ;D)

Mario Agati

DevOps Engineer and Backend developer @ beSharp. Passionate about music, football, and motorsports. In my free time, I like to annoy my neighbors by playing drums.
