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

21 Ottobre 2022 - 9 min. read

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

Uno dei temi che più ci hanno appassionato negli ultimi anni è sicuramente quello dell'**estrazione del valore dal dato**.

Complice anche la grande richiesta - da parte di clienti e non - di progetti data-centrici, ci siamo divertiti a esasperare il concetto di data-driven in molti side projects di R&D arrivando a padroneggiare sempre di più l'argomento.

In beSharp non ci accontentiamo di comprendere. Noi vogliamo sperimentare. Perfezionare. Sporcarci le mani nei contesti più estremi, quelli in cui anche i nerd più accaniti penserebbero che siamo pazzi (true story... ma non ce la prendiamo :D).

E così, qualche tempo fa, in un crescendo di smanettamento sempre più estremo, abbiamo assemblato il primo racing simulator by beSharp: volante direct-drive da 32 Nm di coppia, pedali con smorzatori idraulici progettati per resistere a oltre 140 kg di pressione, sedile racing con cinture a 6 punti e, soprattutto, una piattaforma di movimento a 4DOF per riprodurre la dinamica del veicolo e le asperità del tracciato.

Non contenti, abbiamo costruito pezzo per pezzo anche un **hardware super-carrozzato** che abbiamo subito iniziato a maltrattare installandoci sopra "Assetto Corsa Competizione", uno dei software di simulazione di guida più fedeli mai costruiti.

La parte veramente interessante di questo progetto - nonché la più challenging - è stata però riuscire a intercettare da "Assetto Corsa" **il maggior numero possibile di**

metriche, di portarle efficientemente sul Cloud con un **sistema di ingestion** e, infine, **visualizzarle a schermo in real-time** durante la corsa. Una figata, vero?

D'altro canto, anche Galileo Galiei lo aveva detto: "Misura ciò che è misurabile, e rendi misurabile ciò che non lo è".

Si sa, noi ci ispiriamo solo ai migliori ;)

E' proprio sull'aspetto Cloud della soluzione che ci concentreremo in questo articolo. Vi racconteremo nel dettaglio come abbiamo combinato i servizi di AWS per progetti data-centrici fino ad arrivare ad un risultato soddisfacente secondo i nostri standard.

Cominciamo!

Le sfide tecniche incontrate

L'obiettivo principale da tenere a mente durante la progettazione e lo sviluppo è sempre stato solo uno: le **performance**.

Quindi dovevamo trovare un modo rapido per estrarre tutte le metriche di cui avevamo bisogno dal software che ci permetteva di girare nelle piste più veloci del mondo.

Bene, come punto di partenza avevamo un noto software di simulatore di guida, che utilizziamo per divertirci virtualmente su pista. Da qui nasce la prima sfida tecnica.

Il simulatore di guida utilizzato è infatti pensato per essere installato ed eseguito per giocarci, non per integrarsi con applicazioni terze.

Ma girando nei più remoti spazi di internet, unendo pezzi di documentazione presi qua e là, ci siamo riusciti! Abbiamo scoperto che il software espone un server UDP locale nativamente.

Il protocollo UDP, tra l'altro, faceva al caso nostro, dovendo estrarre dati runtime con performance super ottimali!

Ok perfetto, nulla di più facile, allora creiamo un connettore in Python, estraiamo ed è fatta no? Sarebbe stato bellissimo ma molte metriche mancavano, quindi dovevamo cercare un altra strada.

Non ci siamo fatti abbattere e dopo altre ore di ricerche forse avevamo la risposta: **prenderle direttamente dalla RAM**.

Abbiamo trovato degli indirizzi di memoria che venivano scritti con tutti i dati di cui avevamo bisogno, direttamente dal software del simulatore.

Quindi siamo partiti a sviluppare un piccolo script in Python che estrae i dati binari e li converte in modo da vedere se c'è tutto quello di cui abbiamo bisogno.

Eureka! Questa volta avevamo tutto.

Abbiamo concluso questa prima parte aggiungendo al nostro script di estrazione dati una parte che prende tutti i dati, li impacchetta in JSON e li spedisce in cloud.

Sì, ma dove in Cloud?

L'architettura su AWS

Ingestion

Se siete dei lettori abituali saprete bene che a noi di beSharp fa impazzire la progettazione di infrastrutture cloud altamente performanti, scalabili e affidabili. In questo progetto abbiamo trovato pane per i nostri denti.

La prima cosa alla quale pensare è: quale servizio scegliere come porta di ingresso nel cloud?

Le possibilità a nostra disposizione erano molteplici: potevamo gestire tutto il flusso dei dati proveniente dal simulatore con uno stream Kinesis o con un cluster Fargate ad esempio.

Entrambe le soluzione sarebbero state scalabili e performanti ma queste strade avrebbero portato a dover sviluppare e gestire una componente software che smistasse il traffico in base ai dati in input.

Queste soluzioni non rispettavano pienamente i criteri di qualità infrastrutturale che ci eravamo posti, noi volevamo un infrastruttura interamente Event Driven e Serverless, in grado di garantire un' elevata scalabilità e l'integrazione gestita out of the box con un'ampia gamma di servizi AWS.

Abbiamo trovato queste caratteristiche nel servizio **IoT Core**.

Le caratteristiche che hanno fatto vincere questo servizio rispetto agli altri sono:

- La possibilità di implementare regole, totalmente gestite, in grado di smistare il traffico in base al contenuto che passa dal servizio
- La compatibilità diretta con servizi di analisi realtime

Ok abbiamo capito come far entrare i dati nel nostro account AWS, adesso abbiamo bisogno di dividere questo enorme flusso di dati grezzi in 3 parti:

- Ingestion metriche per Real Time analysis
- Ingestion informazioni circa il tempo del giro
- Ingestion metriche per storicizzazione e future analisi

Ingestion metriche per Real Time Analysis: serve per la visualizzazione real time di tutte le metriche.

I servizi principali che avevamo a disposizione per raggiungere questo obiettivo sono 2: Amazon QuickSight e OpenSearch.

Il nostro scopo era avere un servizio che mostrasse i dati in near real time passanti da una regola creata su IoT core e non fosse da gestire a livello infrastrutturale.

Analizziamo il primo servizio: Amazon QuickSignt, consente di creare una dashboard con numerosi oggetti grafici per una reportistica avanzata.

Il problema fondamentale di questa soluzione è che non è direttamente integrabile con IoT Core.

Bisognerebbe far passare i dati dalla regola creata su IoT Core a un kinesis stream per poi depositare il tutto in un data lake all'interno di Amazon S3.

Solo dopo aver fatto questo potremmo integrarlo con Amazon QuickSight ma in maniera non realtime.

Quindi abbiamo bocciato Amazon QuickSignt con un ottimo livello di certezza, visto che è un servizio pensato per fare business analysis e non realtime analysis.

Passiamo al prossimo candidato: OpenSearch, anche lui ci permette di creare oggetti grafici per report avanzati.

Ma la caratteristica che fa da game changer per questo servizio è l'integrazione diretta e nativa con la nostra regola su IoT Core.

Per creare questa connessione non basta altro che inserire come target, all'interno di una regola, l'endpoint del nostro cluster OpenSearch e l'indice sul quale mandare i dati.

Facile no? Quindi direi che abbiamo un vincitore!

Ingestion informazioni circa il tempo del giro: Il forwarding dei dati, acquisiti dal campo, è stato facilmente implementato usando una regola su IoT Core che seleziona i dati in transito relativi ad un singolo giro e invoca una componente computazionale che prende i dati elaborati e li inserisce all'interno di un servizio di storage.

Quindi per questo flusso dobbiamo scegliere 2 servizi.

Computational Component

Iniziamo dalla parte computazionale: su AWS abbiamo un mare di possibilità per scrivere un software e farlo eseguire in ambiente cloud, ma proviamo a restringere un po' il campo.

Dobbiamo tenere a mente che il nostro requisito è sempre quello di avere un'infrastruttura serverless, estremamente scalabile e che adotti il modello di pagamento Pay As You Use e che sia integrabile direttamente con le regole di IoT Core.

Quindi possiamo già restringere il campo a Fargate e Lambda, i 2 servizi principali per lo sviluppo serverless su AWS.

Il primo servizio ci consente di creare un cluster di container Docker, la parte di scalabilità è da configurare e gestire, inoltre un cluster Fargate non si spegne automaticamente se non viene invocato andando quindi a rincarare sul nostro billing anche quando non siamo in pista.

Quindi non ci resta che utilizzare una Lambda Function, questa a differenza di Fargate rispetta in pieno il concetto di Pay As You Use in quanto, una volta terminato il runtime, il container si spegne, per poi riaccendersi solo alla successiva invocazione.

Una volta scelta la parte computazionale dovevamo scegliere lo storage.

Database

Sicuramente avevamo bisogno di uno storage che fosse facilmente interrogabile per mezzo di query, con performance estremamente elevate, anche questo deve rispettare il concetto di Pay As You Use ed essere totalmente gestito a livello di scalabilità.

Dati questi requisiti quello che abbiamo a disposizione su AWS è Aurora Serverless e DynamoDB.

Il primo ci porterebbe alla configurazione di un cluster che avrebbe sempre un nodo acceso e che quindi non ci farebbe ottimizzare il nostro billing, inoltre non avevamo la necessità di avere dei dati strutturati con delle relazioni.

Quindi abbiamo scelto di utilizzare una singola tabella del servizio DynamoDB che ci fornisce un database NoSQL e key-value per la parte di memorizzazione dei tempi e della leaderboard.

Data Lake

Ingestion metriche per storicizzazione e future analisi: Il terzo ed ultimo flusso è relativo alla parte di DataLake, la nostra soluzione genera una grande mole di dati grezzi che va messa in uno spazio nel quale sarà possibile fare analisi e trasformarli in dati utilizzabili, questo deve essere possibile in qualsiasi momento senza dover ripensare all'infrastruttura.

Quindi la prima domanda a cui dovevamo rispondere era dove conservare i dati.

Volevamo uno storage service scalabile, pay as you use, con la possibilità di suddividere lo spazio con diversi livelli di sicurezza e con diverse lifecycle policy.

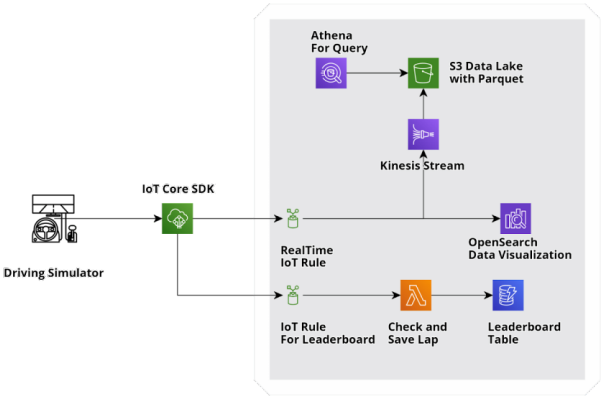
In questo caso la scelta è stata molto semplice, la nostra scelta non poteva che ricadere su Amazon S3 poichè è l'unico servizio all'interno di AWS con tutti i requisiti di cui avevamo bisogno.

Adesso non ci restava altro che capire come mettere i dati sul bucket S3 senza dover leggere da IoT Core e riscrivere sul bucket con un applicativo custom.

Anche in questo caso la scelta è molto semplice, qui è il caso di implementare un Kinesis Stream che grazie alla sua integrazione nativa con Amazon S3 ci consente di

mettere dati sul nostro bucket senza gestirne il trasferimento.

Adesso che abbiamo creato il nostro DataLake, siamo già predisposti per future operazioni di busiiss analysis, grazie a servizi come Athena o QuickSignt che hanno l'integrazione nativa con il servizio S3.



Conclusioni

Vi starete sicuramente chiedendo: "ma vi siete tenuti questo progetto tutto per voi?"

Assolutamente no! Dal primissimo momento in cui abbiamo acquistato il simulatore, avevamo già in mente di fare le cose in grande. Era solo questione di definire la strada (anzi, la pista) da percorrere.

Ed è così che siamo arrivati all'edizione 2022 dell'AWS summit di Milano, l'evento Cloud italiano per definizione, finalmente in presenza dopo due anni di versione virtuale.

Se siete passati di lì, avrete sicuramente percepito il rumore che abbiamo fatto al nostro stand: una Cloud Band che suonava live, poster divertenti, buon caffé e un mare di "viola-beSharp"... ma la star indiscussa è stata sicuramente proprio lui, il Simulatore.

Questa nostra creazione non è stata un semplice punto di partenza, né tanto meno un traguardo; è invece uno dei protagonisti principali di un progetto ambizioso molto più ampio che abbiamo deciso di battezzare come "beSharp Performance Technology", per gli amici "bPT".

bPT è un vero e proprio nuovo brand in cui ci sentiremo liberi di dare sfogo a tutta l'R&D possibile e (IN)immaginabile.

Il perché sia nato, è storia... Anche molto divertente! Per i più curiosi, la trovate qui.

beSharp declina ogni responsabilità per eventuali schizzi di nerditutidine dopo aver letto questo articolo... ma accetta di sapere nei commenti cosa ne pensate di tutto questo folle progetto!

A presto, su Proud2beCloud!

About Proud2beCloud

Proud2beCloud è il blog di beSharp, APN Premier Consulting Partner italiano esperto nella progettazione, implementazione e gestione di infrastrutture Cloud complesse e servizi AWS avanzati. Prima di essere scrittori, siamo Solutions Architect che, dal 2007, lavorano quotidianamente con i servizi AWS. Siamo innovatori alla costante ricerca della soluzione più all'avanguardia per noi e per i nostri clienti. Su Proud2beCloud condividiamo regolarmente i nostri migliori spunti con chi come noi, per lavoro o per passione, lavora con il Cloud di AWS. Partecipa alla discussione!

Paolo Di Ciaula

DevOps Engineer, Frontend Developer e Mobile App Developer @ beSharp. Il mio tempo libero è diviso tra musica, sviluppo... e birra (spesso insieme, per migliori risultati ;D)

Mario Agati

DevOps Engineer e Backend developer @ beSharp. Appassionato di musica, calcio e motori, nel tempo libero mi piace dare fastidio ai miei vicini suonando la batteria.
