# Gestione Proattiva degli Errori Basata sui Dati con AWS X-Ray e CloudWatch

17 Luglio 2024 - 5 min. read

Amazon CloudWatch | AWS X-Ray | Error management

*"L'immaginazione è più importante della conoscenza. La conoscenza è limitata. L'immaginazione abbraccia il mondo."* Albert Einstein

L'esperienza utente è fondamentale per il successo di un sito web: se un utente riscontra errori durante l'utilizzo di un servizio, ciò influenzerà il tasso di conversione ed i guadagni.

Correlare i dati per trovare la causa scatenante di un disservizio è un processo che richiede tempo: spesso è necessario molto lavoro per cercare nell'enorme quantità di log e isolare tra le azioni dell'utente la causa scatenante. A volte è praticamente impossibile identificare i piccoli glitch che peggiorano l'esperienza dell'utente del nostro sito web.

Supponiamo che ad un utente che accede ai nostri servizi web venga assegnato un ID univoco per ogni transazione che compie (come ad esempio un ID carrello, un ID ordine o un cookie di sessione).

Se si verificasse un errore e l'utente ci contattasse per risolverlo, potremmo certamente risalire a cosa stava facendo l'utente e identificare il problema, ma correlando *manualmente* i dati dell'errore all'identificatore utilizzato dall'applicazione.

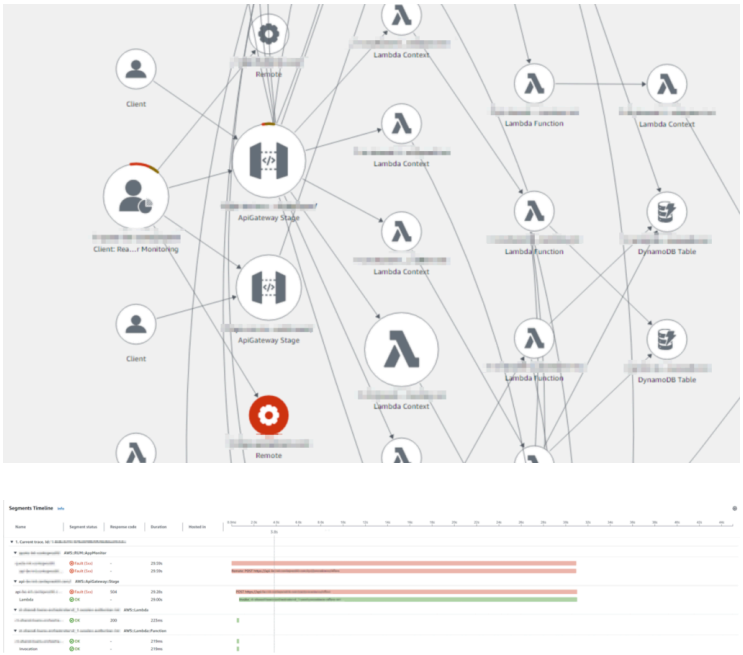
Si tratterebbe di un approccio di tipo *reattivo*.

Come possiamo invece sfruttare *proattivamente* i dati e gli strumenti che già abbiamo per vedere quali transazioni sono impattate da problemi? Possiamo riuscire a contattare facilmente e rapidamente l'utente per aiutarlo prima che si accorga dell'errore e apra un ticket?

Abbiamo già visto soluzioni di monitoraggio su AWS, ad esempio con AWS X-Ray, CloudWatch e Cloudwatch RUM, così come abbiamo visto come estrarre dati significativi per la nostra business intelligence. Tuttavia, **correlare i dati in tempo reale per rispondere rapidamente agli eventi e agli errori** richiede un'integrazione più profonda.

In questo articolo, vedremo come possiamo utilizzare una funzione meno conosciuta di CloudWatch, i **CloudWatch Custom Widgets**, per ottenere informazioni più approfondite sull'esperienza utente senza utilizzare servizi aggiuntivi. Spesso e volentieri, infatti, abbiamo già i dati di cui abbiamo bisogno; dobbiamo solo liberare la nostra immaginazione :)

Prima di iniziare: CloudWatch RUM dispone già di una integrazione con AWS X-Ray: si tratta di attivarla con una semplice checkbox per iniziare tracciare e correlare gli errori con AWS lambda, chiamate API ed i servizi AWS. Ecco un esempio da un caso d'uso reale:




Come si può vedere, con le architetture serverless la complessità cresce ed il tracciamento richiede tempo, pazienza e conoscenza. AWS X-Ray aggiunge un header di tracciamento per facilitare la correlazione tra le richieste degli utenti ed ogni componente; è utile per uno sviluppatore, ma è difficile da usare per altri team come ad esempio team di supporto e analisi dei dati.

Inoltre, non ha informazioni sul dominio applicativo necessarie per prendere decisioni su eventuali modifiche del comportamento dell'applicazione per risolvere problemi e imperfezioni.

Esiste quindi uno strumento che permetta avere un collegamento tra un errore dell'applicazione e i dati dell'applicazione? Certo! **Parliamo di CloudWatch Custom Widgets.**

I Custom Widgets aiutano a ricercare e visualizzare i dati in modo conveniente utilizzando funzioni Lambda ed aggiungendo così una parte di calcolo computazionale che può essere utile ad automatizzare il lavoro di correlazione dei dati disponibili da fonti diverse, minimizzando anche gli errori umani.

Vediamo come costruire una dashboard che visualizzi le transazioni con errori, cercando ed estraendo dati significativi.

Assumeremo per comodità che le trace di AWS X-Ray contengano già l'ID della transazione. Ora occorre  trovare un modo per ottenere tutti gli errori e da qui estrarre i dettagli sulle transazioni e visualizzare i risultati.

Fortunatamente per noi, possiamo cercare in AWS X-Ray gli errori registrati da CloudWatch RUM (che si sono verificati nei browser degli utenti) con una sintassi simile a questa:

```
fault AND (service(id(type: "client::RUM" )))
```

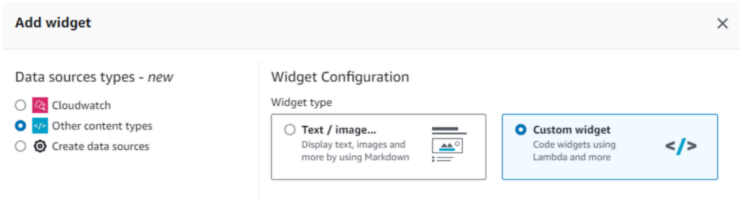
Con questa semplice query otteniamo tutte le tracce.

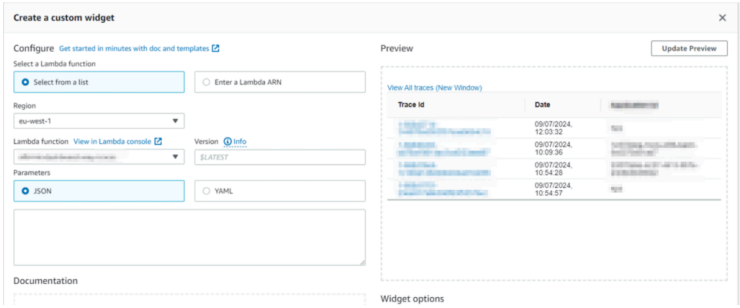
Per ottenere l'ID interno della transazione è possibile usare una funzione lambda che cerchi i record restituiti dalla nostra query utilizzando il nostro linguaggio di programmazione preferito in abbinata alle librerie AWS SDK (ad esempio, python e boto3).

La stessa lambda può cercare altre informazioni nei database, così da poter recuperare dettagli sull'utente, come ad esempio l'email, utili a risolvere il problema.

A questo punto è sufficiente generare codice HTML dalla funzione lambda per visualizzare tutte le informazioni in una dashboard Cloudwatch. A questo indirizzo sono disponibili alcuni esempi.

Una volta sviluppata la lambda, sarà possibile aggiungere un widget personalizzato a una dashboard ed avere l'anteprima del risultato:






In questo modo possiamo avere una visualizzazione che mostri tutti gli errori ed i dati utili al team di supporto, il quale potrà coinvolgere facilmente gli sviluppatori quando necessario.

## Conclusioni e next step

Il nostro caso d'uso è volutamente semplice, ma è possibile espandere questo esempio ed includere maggiori informazioni: pensate al valore aggiunto di avere statistiche su quanto frequentemente si verifica un errore all'interno di un funnel utente specifico.

Una volta trovato un modo per correlare i dati, l'immaginazione è l'unico limite!

Quello descritto è un tipo di approccio simile ai principi della filosofia Unix, che permette di costruire elaborazioni e workflow complesse combinando utility semplici:

- *"Scrivi programmi che fanno una cosa e la fanno bene."*
- "Scrivi programmi che possano lavorare insieme."
- "Scrivi programmi per gestire testo perché è un'interfaccia universale."

Avete avuto difficoltà in passato a correlare i dati e prepararli per mettere a disposizione insight a team non tecnici? Quale soluzione avete adottato? Fatecelo sapere nei commenti!

---

## 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!

**Damiano Giorgi**

Ex sistemista on-prem, pigro e incline all'automazione di task noiosi. Alla ricerca costante di novità tecnologiche e quindi passato al cloud per trovare nuovi stimoli.L'unico hardware a cui mi dedico ora è quello del mio basso; se non mi trovate in ufficio o in sala prove provate al pub o in qualche aeroporto!

---