Perchè è importante l’osservabilità?

di A. Di Blasi, FullStack Developer @ Corley - whitepaper 2020

In un contesto tecnologico come quello attuale, in cui stiamo assistendo al progressivo abbandono di architetture di stampo “monolitico” in favore di applicativi sempre più distribuiti e composti da molteplici componenti, l’utilizzo di strumenti per monitorare e tracciare lo stato dei nostri software è diventato una vera e propria priorità.

L a **Observability è fondamentale** al fine di prevedere qualsiasi possibile imprevisto e rimediare nella maniera più rapida ed efficiente possibile.

Una cosa però va immediatamente sottolineata: *l’osservabilità non è un concetto nuovo*, seppur il suo utilizzo sia diventato imprescindibile con l’avvento dei microservices.

> *Parlando di osservabilità, generalmente intendiamo l’insieme degli strumenti che ci permettono di comprendere in modo granulare il comportamento intrinseco delle nostre architetture*

La presenza di un unico grosso componente, all'interno del quale si posiziona tutta la logica di business come nel caso dei Monoliti, non preclude la possibilità di poter utilizzare tali soluzioni.

Difatti, anche in queste circostanze, la possibilità di poter comodamente visualizzare e interagire con lo stato dei nostri sistemi, oltre ad essere un aiuto essenziale per il team tecnico a cui viene fornito un modo più ingegnoso per il debug della propria struttura, è anche un elemento sostanziale per il business aziendale, che avrà dei riscontri molto positivi sul bilancio operativo: infatti la riduzione dei tempi richiesti per la risoluzione dei problemi si concretizza in **maggior tempo a disposizione** per sviluppare nuove features.

I tre pilastri dell'osservabilità

Observability, tracing, logging, metrics, monitoring.

Questi termini spesso generano confusione e si tende erroneamente a pensare che in realtà rappresentino gli stessi concetti. **Niente di più sbagliato!** Parlando di osservabilità, generalmente intendiamo l'insieme degli strumenti che ci permettono di comprendere in modo granulare il comportamento intrinseco delle nostre architetture. Per poter definire i nostri sistemi come "Osservabili", è infatti necessaria l'applicazione dei cosiddetti tre Pilastri dell' Osservabilità: le *metriche*, i *logs* e il *tracing*, senza i quali un software non può essere definito davvero Osservabile.

Attenzione però: non è un obbligo l'adozione di tutti i pilastri, in quanto, seppur essi si completino a vicenda, l'utilizzo anche solo parziale può rivelarsi efficace e conveniente.

In un caso pratico possiamo immaginare l'integrazione di sistemi di tracing per **visualizzare il flusso** dei componenti coinvolti in ciascuna operazione svolta da un utente.

Questo produrrà un risultato considerevole e sarà visualizzabile e integrabile, indipendentemente dal fatto che parallelamente sia presente

un modo per visualizzare l'utilizzo di sistema dei nostri microservizi (Metriche).

Le metriche

Le metriche ci forniscono un metodo per avere una panoramica efficace relativamente allo stato dei nostri sistemi, siano questi macchine fisiche on premise, istanze in cloud o containers. Solitamente le metriche più comuni raggruppano dati come:

- l'utilizzo del disco e della CPU
- il consumo di banda
- il consumo della memoria.

Molto utili risultano anche le metriche applicative, attraverso le quali possiamo ottenere informazioni rilevanti a livello software: pensiamo ad esempio, all'utilità di poter visualizzare e interrogare alcuni dati fondamentali come l'insieme di errori HTTP generati dai nostri servizi web in un dato periodo di tempo; in questo modo siamo in grado di avere una visione sul corretto funzionamento degli applicativi, con l'aggiunta di potersi eventualmente integrare a sistemi di **notifica in tempo reale** nell'eventualità che alcune delle nostre metriche superino delle soglie definite in precedenza.

Ad esempio potremmo decidere di notificare una o più persone nel caso la percentuale di utilizzo della cpu in un dato periodo di tempo superasse l'80%, così da poter effettuare quanto prima operazioni con l'obiettivo di evitare, o perlomeno limitare, i danni che porterebbero ad una completa irraggiungibilità del nostro applicativo.

Logging

I log sono dei record identificati dal timestamp in cui sono registrati: essi ci garantiscono una **visualizzazione dettagliata dello stato della nostra applicazione ad una data ora**, consentendoci di navigare tale stato sulla base di un asse temporale.

Sicuramente il logging rappresenta il pilastro più popolare e diffuso nei

contesti di monitoring. Tutti gli applicativi generano continuamente dei log: Database, API, sistemi di cache... Data la spaventosa mole di dati da gestire, l'adozione di sistemi di logging si direzione più che sull'effettiva produzione di tali dati, sulla capacità di saperli sfruttare e utilizzare con l'obiettivo di renderli utili per i nostri fini di Business.

Tracing

Per ultimo, abbiamo il pilastro che potrebbe considerarsi frutto delle attuali architetture distribuite, in cui molti piccoli componenti collaborano al fine di costruire la nostra struttura, ovvero il tracing. Esso è la rappresentazione del **flusso end-to-end che attraversa l'insieme dei microservizi** coinvolti in ciascuna operazione.

Capire quanto tale strumento sia straordinario non richiede troppo sforzo. Immaginando una situazione reale basata su microservizi e priva di una soluzione di tracing, nel caso di latenza del sistema, l'unica soluzione sarebbe verificare, microservizio per microservizio, quale dei nostri componenti sia il collo di bottiglia causa del problema diffuso.

E' evidente che un approccio di questo tipo richiederebbe un dispendio di tempo (e di conseguenza un aggravio economico) molto rilevante per le aziende, che non tutte sarebbero in grado di affrontare.

Ecco come in questo caso il tracing ci verrebbe in aiuto, permettendoci di identificare e sistemare rapidamente il componente interessato dal problema, così da ottimizzare tempi e risorse.

C'è sempre bisogno di osservabilità?

È una convinzione comune che *l'osservabilità sia utile sono all'interno di progetti molto grandi*; possiamo però considerarla una verità? Né sì né no... Se da un lato chiaramente l'approccio a strumenti di questo tipo richiede una discreta mole di tempo per essere integrata e utilizzata, cosa che in alcuni progetti troppo piccoli potrebbe non essere giustificabile, dall'altro lato non preclude l'**integrazione parziale** di solo alcuni aspetti dell' osservabilità che abbiamo discusso, rimandando gli

altri al momento in cui si riveleranno utili al fine di ottimizzare le nostre risorse.

Nel caso di un sistema non distribuito, per esempio, potrebbe rivelarsi inefficace l'approfondimento di soluzioni di tracing in una fase iniziale, a favore di soluzioni basate su metriche e logging.

Cosa significa implementare un sistema di observability?

Nel concreto, l'attuazione delle tematiche discusse è abbastanza immediata. Chiaramente il tempo di integrazione dipende dal grado di granularità delle informazioni a cui siamo interessati. Comunemente per quanto concerne il logging o la cattura delle metriche, vi è un *Agente* inserito all'interno delle delle istanze di cui si desidera ottenere i dati.

Esso svolge automaticamente le operazioni di ottenimento e aggregazione dei valori sulla base di un file di configurazione all'interno del quale vengono definiti i possibili *Input* (ovvero le sorgenti da cui attingere per l'ottenimento delle informazioni) e degli *Output* (cioè dove si intende inviare tali dati).

Una volta preparati e inviati i risultati verso uno o più di questi output, sarà necessario un modo semplice ed efficiente per interrogare e visualizzare tali dati.

Avremo quindi la necessità di **specifiche dashboard** che garantiscano la possibilità di rendere tali informazioni accessibili a chiunque, anche alle figure meno tecniche, le quali, pur non essendo interessate ai dettagli di implementazione, potrebbero però avere un'idea dello stato generale del sistema.

Una soluzione molto diffusa per l'adozione di tali componenti è l'Elastic Stack (https://www.elastic.co/), il quale comprende diversi tools che integrano tutte le esigenze discusse, nello specifico:

- Logstash per la cattura e aggregazione di logs e metriche
- Elastic Search come output per la conservazione dei dati ottenuti

- Kibana, per la dashboard di visualizzazione per i risultati conservati su Elastic Search

Conclusione

In conclusione, possiamo affermare che, al giorno d'oggi, **l'osservabilità non può continuare ad essere ritenuta un'opzione**: essa rappresenta infatti l'unico modo per una gestione coerente e logica dei nostri applicativi, senza la quale il debug e la verifica dei nostri sistemi si traducono in un brancolare nel buio alla ricerca del problema, che ovviamente è un'operazione difficile, se non impossibile da perseguire sul lungo periodo.

> *L'Osservabilità rappresenta l'unico modo per una gestione coerente e logica dei nostri applicativi*

Perciò non è importante complicare la propria struttura cercando il quanto più possibile di renderla adatta ad essere osservata, ma *capire le specifiche esigenze*, in modo da integrare solo ciò che è funzionale a ottimizzare la propria situazione dal punto di vista tecnico ed economico, garantendo agli Sviluppatori e agli Amministratori un "modus operandi" più sereno e proficuo.