Da prodotto a servizio: i microservizi come building block per facilitare la servitizzazione

13 Marzo 2024 - 8 min. read

Decoupled Architectures *Microservices* *Servitization*

Nel mondo dinamico del cloud computing, dove agilità e scalabilità sono fondamentali, il disaccoppiamento dei servizi rappresenta una pietra miliare per la creazione di sistemi resilienti e adattabili. In questo blog, abbiamo approfondito vari aspetti tecnologici della servitizzazione, esplorando strategie per il disaccoppiamento dei microservizi, modelli di comunicazione asincroni e architetture disaccoppiate che sfruttano event busses.

Questo articolo mira a sintetizzare la principali informazioni, fornendo una prospettiva completa sull'importanza del disaccoppiamento dei servizi e sulla sua relazione con la servitizzazione.

Senza ulteriori indugi, cominciamo il nostro viaggio!

Servitizzazione e architetture basate su microservizi

La servitizzazione implica la trasformazione dei prodotti in servizi, consentendo alle organizzazioni di offrire valore ai clienti attraverso esperienze più flessibili e personalizzate.

La **servitizzazione riguarda più il business che la tecnologia**; tuttavia, è strettamente legato ai microservizi come modello architettonico principale nelle architetture cloud.

Infatti, costruire un servizio di successo sta diventando sempre più impegnativo a causa delle aspettative molto elevate dei clienti e degli standard di mercato di conseguenza elevati. Costruire un servizio di qualità richiede, oggi più che mai, tempo e impegno ed è spesso complesso, soprattutto se la user base è ampia e geograficamente distribuita.

È qui che i microservizi tornano utili.

L'essenza del paradigma dei microservizi risiede nella **scomposizione di applicazioni complesse in componenti più piccoli e autonomi** (microservizi) che operano in modo indipendente ma collaborativo per soddisfare varie funzioni aziendali. Questo approccio migliora la modularità e la flessibilità del sistema e facilita il perfetto adattamento alle mutevoli richieste del mercato e alle esigenze dei clienti.

Uno dei principali vantaggi derivanti dall'adozione di un'architettura basata su microservizi è la sua flessibilità intrinseca. Suddividendo le applicazioni in servizi distinti e autonomi, diventa possibile modificare, aggiornare e scalare i singoli componenti senza interrompere l'erogazione del servizio. Questa modularità consente una rapida innovazione e iterazione, consentendo ai team di rispondere rapidamente alle dinamiche del mercato e al feedback dei clienti.

La scalabilità è un altro vantaggio significativo dell'approccio ai microservizi. A differenza delle architetture monolitiche, in cui la scalabilità implica spesso la replica dell'intero stack applicativo, i microservizi consentono una scalabilità granulare che ottimizza l'utilizzo delle risorse e migliora le prestazioni e la reattività del sistema in condizioni di carico di lavoro imprevedibile.

Inoltre, l'isolamento dei guasti insito nelle architetture dei microservizi migliora la resilienza del sistema e la tolleranza agli errori. In un sistema monolitico, un guasto in un componente può ripercuotersi sull'intera applicazione, determinando tempi di inattività diffusi o interruzioni del servizio. Al contrario, con i microservizi, i guasti dovrebbero essere contenuti all'interno dei singoli servizi, impedendo loro di interrompere completamente l'esperienza dell'utente e minimizzando l'impatto complessivo sulla disponibilità del sistema.

Le architetture a microservizi facilitano inoltre le pratiche di sviluppo agile e il continuous deployment. Poiché ciascun servizio opera in modo indipendente e potenzialmente gestito da un team dedicato, è più semplice adottare la metodologia agile e le pratiche DevOps per accelerare i cicli di sviluppo e semplificare i processi di rilascio. Questa agilità consente di seguire più prontamente i feedback degli utenti e quindi fornisce un vantaggio competitivo. In aggiunta, permette di accelerare l'innovazione riducendo o addirittura annullando il costo del fallimento e quindi permettendo sperimentazioni.

Inoltre, la natura modulare dei microservizi semplifica l'integrazione con servizi di terze parti e sistemi esterni. Il disaccoppiamento dei servizi e la definizione di interfacce chiare facilitano l'integrazione di nuove funzionalità con un impatto minimo sui sistemi esistenti, promuovendo l'innovazione e la collaborazione in tutto l'ecosistema.

Naturalmente, la creazione di un'applicazione basata su un'architettura a microservizi può anche presentare sfide rispetto alle tradizionali applicazioni monolitiche.

La sfida più grande è la sicuramente rappresentata dalla **maggiore complessità nella gestione di più servizi e delle loro interazioni**. A differenza delle applicazioni monolitiche, in cui tutti i componenti sono strettamente integrati, in una buona applicazione a microservizi questi sono loosely coupled, il che richiede un'attenta orchestrazione e coordinamento.

Vi è poi un aumento dell'effort di operation legato al maggior numero di parti funzionanti. Ogni componente viene generalmente eseguito nel proprio container/istanza/FaaS, determinando una proliferazione di risorse da gestire e aumentando la complessità operativa in termini di distribuzione e monitoraggio.

Inoltre, il debug e la risoluzione dei problemi possono essere più impegnativi in un ambiente di microservizi. Con più servizi che interagiscono, identificare e diagnosticare i problemi può essere complesso, soprattutto quando gli errori si propagano oltre i confini del singolo servizio.

Nonostante queste sfide, i vantaggi dei microservizi spesso superano le difficoltà, rendendo questo paradigma sempre più rilevante.

Decoupling

Il disaccoppiamento (decoupling) gioca un ruolo fondamentale in una buona applicazione basata su microservizi. Significa progettare componenti all'interno di un sistema affinché funzionino in modo indipendente, con una dipendenza minima o nulla dal funzionamento interno degli altri. Questo principio consente un migliore isolamento degli errori e maggior scalabilità granulare.

Nei sistemi sincroni, dove i componenti interagiscono in tempo reale, strumenti come i bilanciatori di carico svolgono un ruolo cruciale nel disaccoppiamento. I sistemi di bilanciamento del carico distribuiscono il traffico in entrata su più istanze/container di un servizio, consentendo la scalabilità orizzontale e garantendo che nessun singolo componente diventi un collo di bottiglia. Questo meccanismo di disaccoppiamento sincrono consente alle organizzazioni di gestire carichi di lavoro variabili senza compromettere le prestazioni o l'affidabilità.

D'altro canto, i sistemi asincroni si affidano alla comunicazione basata su messaggi per ottenere il disaccoppiamento. Code, bus di eventi e sistemi di notifica sono componenti

critici nelle architetture asincrone, poiché facilitano l'accoppiamento lasco e l'isolamento dei guasti.

Le code fungono da intermediari tra produttori e consumatori, memorizzando nel buffer i messaggi e consentendone l'elaborazione successiva. Disaccoppiando i tempi di produzione e consumo dei messaggi, le code consentono un utilizzo più efficiente delle risorse e un funzionamento del sistema più fluido.

I bus degli eventi rappresentano la spina dorsale delle architetture distribuite e consentono una comunicazione continua tra i servizi senza accoppiamento diretto. Forniscono un modello di pubblicazione-sottoscrizione in cui i produttori pubblicano eventi su un bus centrale e i consumatori si iscrivono a specifici tipi di eventi di interesse. Questo modello di comunicazione disaccoppiato consente maggiore flessibilità ed estensibilità, poiché è possibile aggiungere o rimuovere nuovi servizi senza interrompere il sistema complessivo.

Allo stesso modo, i sistemi di notifica come Amazon SNS (Simple Notification Service) consentono la comunicazione asincrona tra i componenti, consentendo aggiornamenti e avvisi in tempo reale. Disaccoppiando il mittente e il destinatario dei messaggi, i sistemi di notifica consentono architetture più resilienti e reattive, in cui i componenti possono reagire dinamicamente ai cambiamenti ambientali.

Serverless

Nel panorama in continua evoluzione del cloud computing, le architetture serverless rappresentano un punto di svolta, offrendo un cambio di paradigma nella creazione, implementazione e gestione dei sistemi distribuiti. In prima linea in questa rivoluzione ci sono servizi come AWS Lambda e Fargate che forniscono potenza di calcolo alle applicazioni basate sui microservizi con affidabilità e scalabilità integrata.

AWS Lambda consente agli sviluppatori di eseguire codice senza effettuare il provisioning o gestire i server. Invece di preoccuparsi dell'infrastruttura, gli sviluppatori possono concentrarsi sulla scrittura del codice e sulla definizione degli eventi che ne attivano l'esecuzione. Questo approccio serverless offre scalabilità e flessibilità senza pari, poiché Lambda si ridimensiona automaticamente per gestire (quasi) qualsiasi carico di lavoro.

Tuttavia, i servizi serverless possono essere utilizzati anche per creare il livello di disaccoppiamento delle moderne applicazioni a microservizi.

Amazon SQS (Simple Queue Service), Amazon SNS (Simple Notification Service) e Amazon EventBridge sono fondamentali per il disaccoppiamento.

SQS fornisce un servizio di accodamento dei messaggi affidabile e scalabile, consentendo la comunicazione asincrona tra le diverse parti del sistema.

D'altro canto, SNS abilita la messaggistica pub/sub, consentendo ai componenti di comunicare in tempo reale senza accoppiamento diretto.

Amazon EventBridge funge da bus di eventi centrale, facilitando l'integrazione e l'orchestrazione di eventi tra diversi servizi e sistemi. EventBridge consente agli sviluppatori di definire regole e trigger per l'elaborazione degli eventi, consentendo una comunicazione e un coordinamento senza soluzione di continuità tra componenti disparati.

Oltre alla scalabilità e all'affidabilità, l'architettura serverless offre altri vantaggi, come facilitare cicli rapidi di sviluppo e CI/CD. Con Lambda, gli sviluppatori possono eseguire rapidamente l'iterazione del proprio codice, implementando le modifiche in pochi secondi anziché in giorni o settimane.

Questa agilità consente alle organizzazioni di rispondere rapidamente ai mutevoli requisiti aziendali e alle condizioni di mercato, offrendo loro un vantaggio competitivo nel frenetico panorama digitale di oggi.

Conclusioni

In uno scenario digitale che va sempre più nella direzione della servitizzazione, il successo di un business è sempre più influenzato dalle prestazioni dei servizi erogati, piuttosto che dalla vendita di prodotti. Fornire servizi digitali di alta qualità e rispettare gli standard e le aspettative dei clienti riguardo alla disponibilità, alla reattività e all'esperienza utente complessiva richiede soluzioni tecnologiche che consentano sviluppo rapido e resilienza del servizio.

L'adozione di tecniche di decoupling e di approcci asincroni, così come dei servizi serverless e delle tecnologie Cloud-native permette di raggiungere questo obiettivo.

Se da un lato dunque è indubbio il ruolo fondamentale dei microservizi in un percorso di servitizzazione di successo, è bene ricordare che non si può prescindere dalla capacità di controllare e gestire questo cambio di paradigma.

Speriamo che in nostro articolo abbia contribuito a chiarire le idee in questo senso.

Questo è tutto per l'articolo di oggi; restate sintonizzati per altri contenuti entusiasmanti!

Risorse

- Comunicazione asincrona nei sistemi distribuiti: best practices e impatto su prestazioni e affidabilità
- Disaccoppiare servizi con SQS e Lambda trigger
- Un tuffo nelle architetture cloud disaccoppiate con gli Event Bus

---

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!

Alessio Gandini

Cloud-native Development Line Manager @ beSharp, DevOps Engineer e AWS expert.Computer geek da quando avevo 6 anni, appassionato di informatica ed elettronica a tutto tondo. Ultimamente sto esplorando l'esperienza utente vocale e il mondo dell'IoT.Appassionato di cinema e grande consumatore di serie TV, videogiocatore della domenica.
