SCOMPORRE UN'APPLICAZIONE MONOLITICA LIKE A PRO: LA NOSTRA GUIDA AD UNA TRANSIZIONE (QUASI) SENZA SFORZO.

Microservices | Software as a Service (SaaS)

beSharp | 24 Maggio 2019

INTRODUZIONE

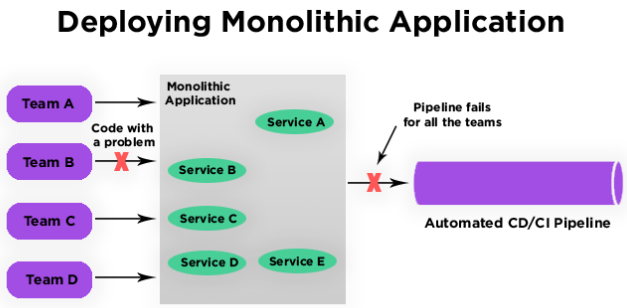
Man mano che le applicazioni software tendono a crescere sempre di più, aggiungendo nuove funzionalità, definendo logiche sempre più complesse, molte interazioni e accoppiamenti forti tra le loro componenti, diviene sempre più difficile gestire il loro sviluppo e la loro manutenzione spingendo quindi le aziende a suddividerle in piccole parti riutilizzabili costruite attorno alle **business capabilities**, i microservizi.

All'interno di un'architettura a microservizi, ogni componente applicativo esiste e opera in maniera autonoma come un singolo servizio comunicando con gli altri servizi tramite delle API ben definite rispettando il **Principio di Singola Responsabilità**.

Migrare verso un **ecosistema a microservizi** non è un viaggio semplice ma è sicuramente uno che vale la pena affrontare; intraprendere tale viaggio significa dare alla vostra applicazione la possibilità di **crescere più velocemente**, **ridurre i rischi di modifica del codice** e gli **alti costi di manutenzione** ad essa correlati.

Immaginate che più di un gruppo di sviluppatori lavori su un'applicazione monolitica, errori su una logica sticky (con un alto tasso di correlazione) possono potenzialmente bloccare il lavoro degli altri team, in quanto di solito la pipeline di CD/CI è la stessa per tutti i membri del progetto; con un approccio a microservizi ogni team può lavorare e mantenere il proprio codice.

Questo approccio è inoltre molto utilizzato nel paradigma SaaS, poichè il modello di rilascio **Software as a Service** si basa sul centralizzare nel Cloud l'hosting delle applicazioni che vengono accedute tramite Thin Client (senza particolare logica operativa) o più comunemente tramite Browser Web, perciò un ecosistema a microservizi asserve a questo scopo molto bene.



Viene data ai vostri team la possibilità di **crescere in numero** e di **dividersi e concentrarsi su ogni singola feature in parallelo**, diventando più efficienti; inoltre il **costo per introdurre un nuovo membro nel team in termini di conoscenza e comprensione del codice è notevolmente ridotto**.

Anche i **costi delle sperimentazioni** sono notevolmente ridotti, in quanto ogni feature è **atomica** e quindi i **cambiamenti non possono danneggiare altre parti dell'applicazione**. Questo permette ai team di generare **valore aziendale** più velocemente.

Ogni microservizio può essere scritto **utilizzando diversi framework** e **linguaggi di programmazione,** e può essere **distribuito in modo indipendente**; grazie alla loro natura atomica e al fatto che aderiscono al **principio di isolamento**, i microservizi garantiscono **soluzioni di disaster recovery** più semplici e veloci.

INIZIAMO IL NOSTRO VIAGGIO

Poiché l'argomento è molto complesso e vasto, cercheremo di concentrarci su tecniche e consigli generali ben noti e non entreremo nel dettaglio di linguaggi di programmazione o infrastrutture specifiche.

Cominciamo dicendo che "micro" nel termine microservizi è solo un'**etichetta** "non" una **descrizione**, nel senso che non c'è bisogno di cercare di dividere la propria applicazione in una molteplicità di piccoli (e potenzialmente inutili) servizi sul cloud, ma afferrare l'idea che si "ha davvero bisogno" di trovare e suddividere **parti ben note del proprio codice capaci di fare operazioni molto specifiche**, con un **elevato valore** e con un **dominio chiaro** e **comprensibile**.

PREREQUISITI

Anche se lavorare con i microservizi può essere estremamente interessante per gli sviluppatori, la vostra azienda deve avere un certo grado di preparazione per poter intraprendere questo viaggio di migrazione al paradigma dei microservizi.

Senza dubbio gli sviluppatori devono avere una chiara comprensione, e idealmente una certa esperienza di lavoro, su tecnologie come Containerizzazione su Docker, Kubernetes ed eventualmente AWS Lambda per ospitare i vostri microservizi.

Bisogna essere in grado di rendere operativo un servizio nel minor tempo possibile, in modo da poter definire le pipeline di **Continuous Deployment** e **Continuous Integration**.

Infine è necessario essere in grado di fornire strumenti di monitoraggio adeguati per ispezionare rapidamente il vostro ecosistema a microservizi (in AWS, ad esempio, un uso intensivo di strumenti come X-Ray, CloudWatch, CloudTrail ed ElasticSearch con Kibana).

Questi prerequisiti richiedono che la vostra azienda segua la **DevOps Culture**, in cui ogni membro del team deve avere una chiara comprensione sia dei **compiti ad appannaggio degli sviluppatori** che dell'**ambiente operativo su cui il codice andrà rilasciato**.

SUDDIVIDERE I LIVELLI APPLICATIVI

Questo è di sicuramente il primo passo e probabilmente il più semplice in termini di comprensione di come può essere realizzato. Quando si inizia a "spezzare" un monolite un task molto semplice è individuare aree che possono essere naturalmente disaccoppiate come ad esempio **front-end** e **back-end**, perché idealmente il front-end dovrebbe essere visto come un semplice consumatore di API; un esempio è dato da un'applicazione sviluppata in Rails: le pagine html sono solitamente renderizzate a partire da **template .erb** attualizzati tramite dati **serviti dal back-end Rails** e il contenuto è altamente accoppiato con esso. Creare una Single Page Application con Angular, ad esempio, e farla comunicare con un back-end **attraverso le API**, permette di disaccoppiare naturalmente i due livelli dell'applicazione. Questo processo è ovviamente potenzialmente lungo in termini di tempo di sviluppo, ma è molto importante, quindi sforzatevi sempre di raggiungerlo e può essere, in certa misura, anche parallelizzato.

Per ora possiamo concludere qui la nostra analisi in quanto abbiamo visto i punti di forza di un approccio a microservizi e come il vostro team e, in generale, la vostra azienda deve essere preparata ad avviare questo lungo ma possibilmente fruttuoso processo. Restate sintonizzati per la seconda parte, in cui inizieremo a descrivere passo dopo passo le strategie per completare la migrazione da Monolite ad un ecosistema a microservizi.

A presto con la seconda parte!

Leggi la seconda parte >>

beSharp

Dal 2011 beSharp guida le aziende italiane sul Cloud. Dalla piccola impresa alla grande multinazionale, dal manifatturiero al terziario avanzato, aiutiamo le realtà più all'avanguardia a realizzare progetti innovativi in campo IT.

Get in touch

beSharp.it
proud2becloud@besharp.it