SPARKFABRIK

eBook

SRE per applicazioni Cloud Native

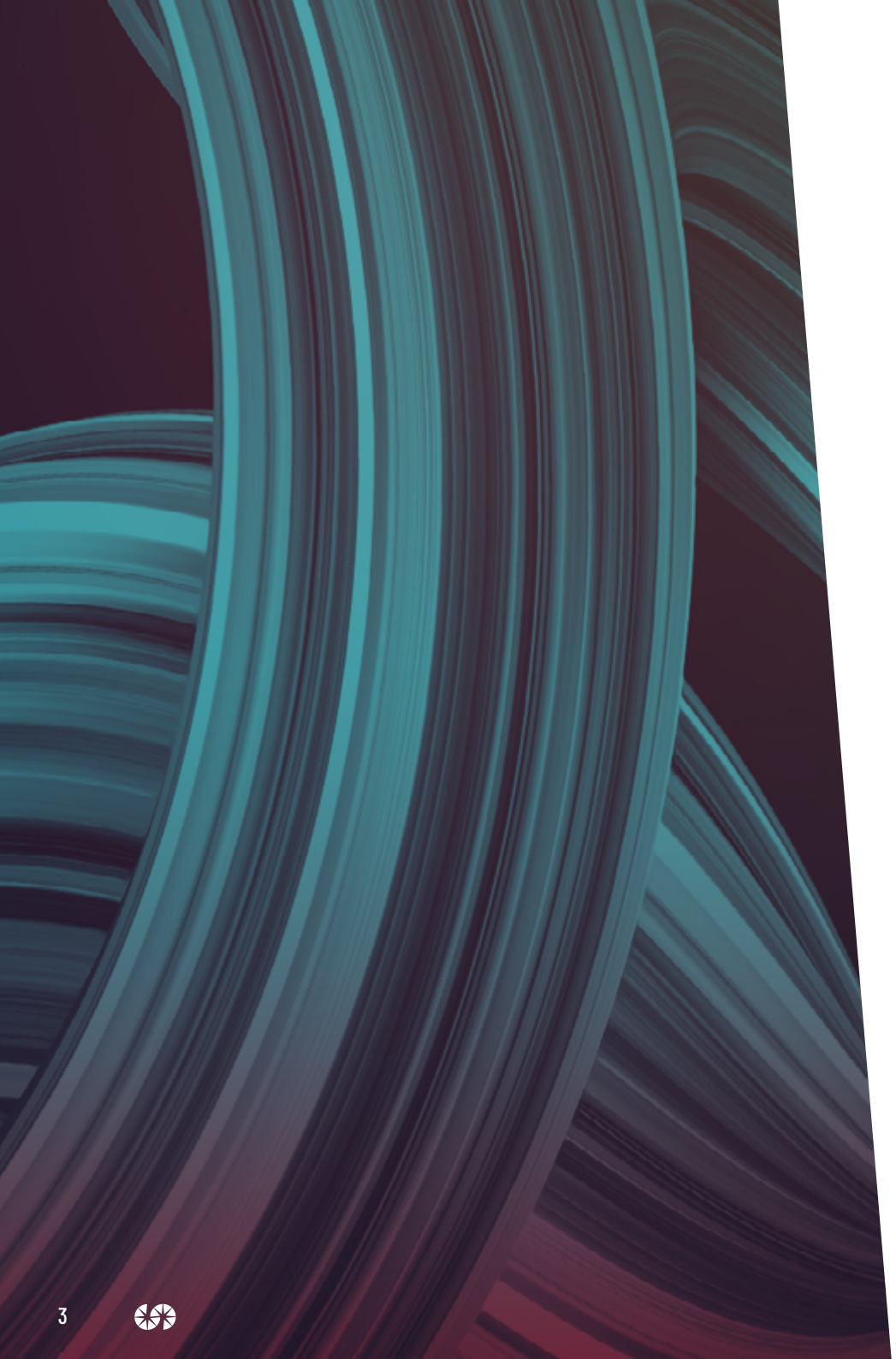
Introduzione

I grandi del tech, come Amazon, Google, Microsoft, sono diventati tali anche perché sono stati capaci di **innovare con successo i propri modelli di sviluppo, organizzativi e tecnologici**. Per seguire il sentiero tracciato da loro sono necessari una strategia corretta e i giusti strumenti metodologici.

In un contesto di business come quello attuale, in piena evoluzione, le aziende sono alla costante ricerca di un equilibrio tra il bisogno di portare avanti con profitto le proprie attività e quello di rispondere ai cambiamenti della clientela e del mercato in modo rapido ed efficace. Ed è qui che l'innovazione in ambito applicativo può fare davvero la differenza: ad esempio, **grazie alla Cloud Transformation, all'adozione dell'approccio Cloud Native e DevOps**, le organizzazioni possono migliorare notevolmente **il time-to-market, la user experience e i servizi digitali offerti**.

L'errore da non commettere è pensare di potersi concentrare solo sulla manutenzione e non anche sull'evoluzione dell'applicazione. **Il cambiamento è inevitabile**: per fare in modo che sia anche sostenibile è necessario essere in grado di poter cambiare in modo sicuro tutto quello che serve a far evolvere le proprie applicazioni. E bisogna poterlo fare per tutta la durata del ciclo di vita dei prodotti. **La Site Reliability Engineering (SRE)**, in questo senso, riveste un ruolo fondamentale.

Indice



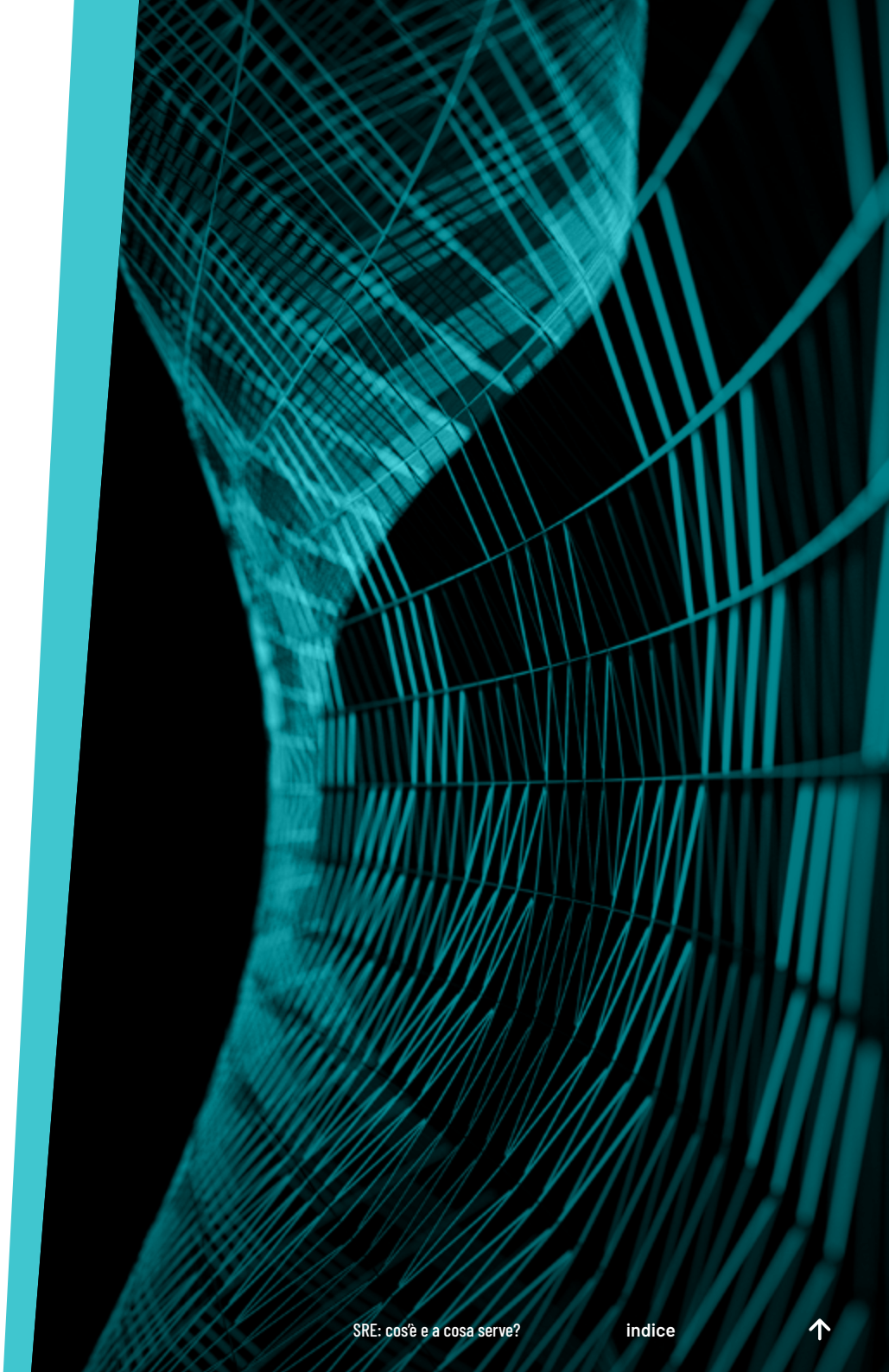
SRE: cos'è e a cosa serve?

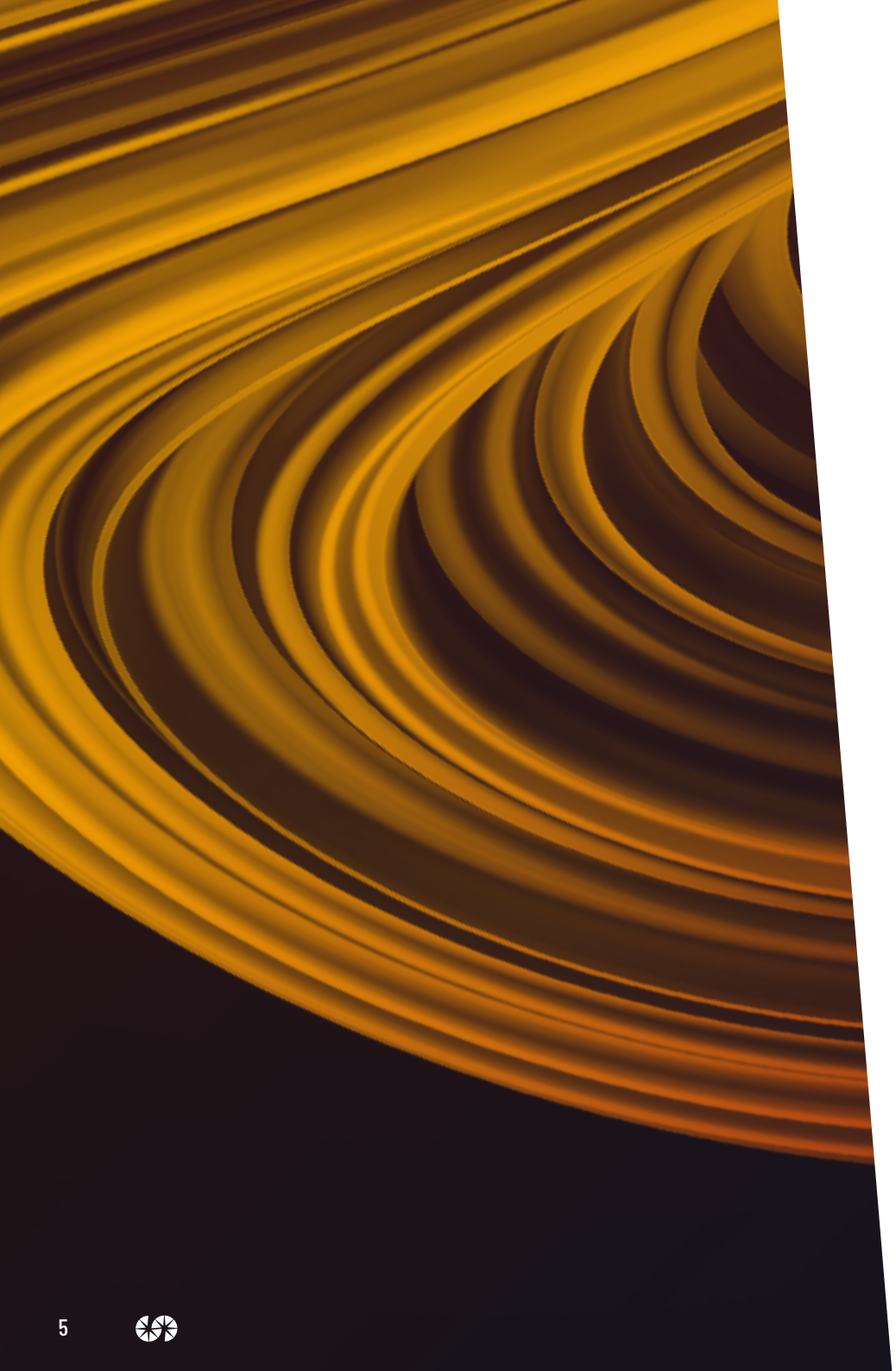
Si tratta di **un termine coniato dagli ingegneri di Google per spiegare come vengono gestiti internamente i loro sistemi**. La SRE, in estrema sintesi, affronta le problematiche di gestione delle infrastrutture e delle operazioni IT applicando l'ingegneria del software.

SRE vuol dire essere capaci di rispondere in tempi rapidi a eventi imprevedibili documentando e automatizzando procedure e soluzioni, in maniera tale da investire più tempo per sviluppare nuove funzionalità e far evolvere gli applicativi.

In questo modo si stabiliscono obiettivi a livelli eccellenti di performance e affidabilità delle applicazioni, prendendosi cura del loro funzionamento senza tralasciare di lavorare a una loro costante evoluzione.

Con la SRE la responsabilità delle applicazioni dell'azienda è condivisa tra il team di prodotto e il team SRE, che si occupa dell'affidabilità. **Questo approccio prevede l'uso di strumenti per la gestione automatica delle operazioni IT** volti tanto a ottimizzare l'affidabilità attuale quanto a migliorare progressivamente l'applicazione.





SLI, SLO e SLA: cosa sono e perché sono il cuore del metodo SRE

Al centro del metodo SRE ci sono tre sigle: **SLI, SLO e SLA**. Non sono termini che servono a creare una documentazione legale per il contratto di servizio, ma piuttosto concetti fortemente legati all'operatività. **Sono pensati infatti per allineare tutti gli attori sugli obiettivi di affidabilità e disponibilità del servizio e sulla sua performance**.

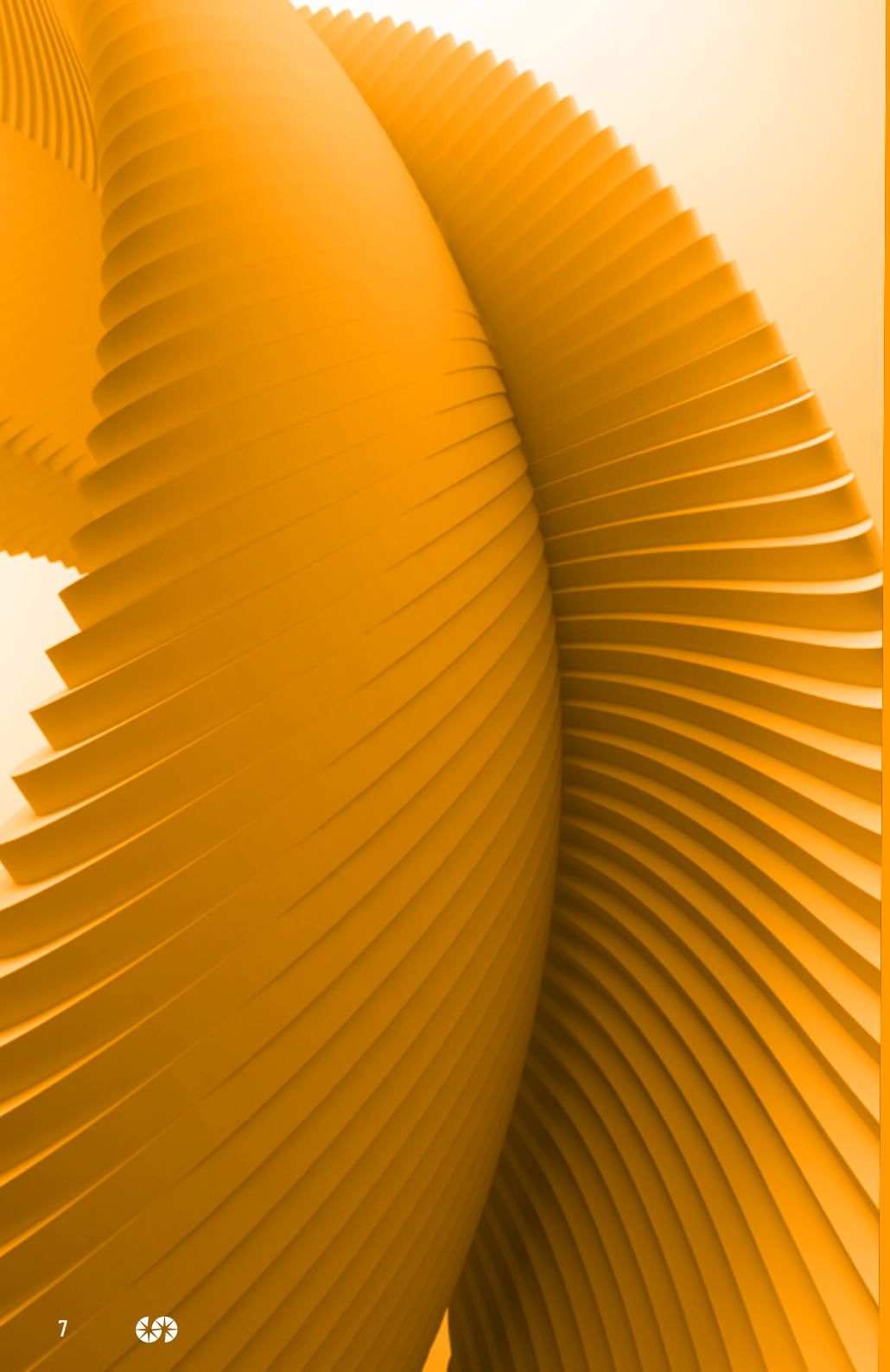
Per poter fissare in maniera efficace degli obiettivi è necessario capire bene tutti e tre i termini: vediamoli uno per uno.

SLI: Service Level Indicators

Per prima cosa parliamo dei **Service Level Indicators (SLI, indicatori di livello di servizio)**, ovvero la metrica con la quale si misura un aspetto della performance o in generale del comportamento dell'applicazione. Solitamente si tratta di **misure relative ad alcune caratteristiche importanti per gli utenti**, come il tempo di risposta delle applicazioni web, il tasso di errore e altro.

Un monitoraggio sintetico e sistematico del funzionamento del sistema è necessario perché consente di avere **una visione d'insieme** significativa non solo per il team SRE ma anche per gli utenti dell'applicazione sia all'interno dell'azienda che al suo esterno.

Questa attenzione ai singoli parametri che gli utenti usano per valutare i servizi e le applicazioni in esecuzione è un aspetto fondamentale dell'approccio SRE. Anziché monitorare semplicemente il tempo di funzionamento del processo, cioè il suo uptime complessivo, **gli SLI incoraggiano la qualità del lavoro** che è, in ultima analisi, uno dei più importanti criteri per definire il successo di un'applicazione.



SLO: Service Level Objectives

Una volta definite le metriche, si possono definire anche gli obiettivi per ciascuna di esse. Qui entrano in gioco i **Service Level Objectives (SLO)**, cioè gli obiettivi di livello di servizio per l'applicazione che viene gestita dal team SRE. Il punto alla base degli SLO è la **definizione di un livello di performance o disponibilità del servizio** che sia all'altezza delle aspettative di utilizzo dal punto di vista del business, e che contemporaneamente abbia anche un costo accettabile.

SLA: Service Level Agreement

Infine abbiamo il **Service Level Agreement, cioè l'accordo sui livelli di servizio o SLA**. Se gli SLO rappresentano l'obiettivo da raggiungere attraverso la misurazione dei Service Level Indicators, **lo SLA è la promessa che viene fatta per raggiungere quell'obiettivo**. Inoltre, lo SLA comprende anche l'insieme degli aspetti legali che definiscono le conseguenze se il sistema non raggiunge i suoi SLO, **incluso l'error budget**, cioè il quantitativo massimo di tempo in cui un sistema può non funzionare senza avere delle conseguenze contrattuali.

L’importanza dell’error budget

Il lavoro del team SRE **non è solo quello di automatizzare per quanto possibile le operazioni in maniera tale che i ticket vengano risolti in modo più efficiente e veloce**. Questa è la prima metà.

La seconda metà del lavoro consiste nella capacità di bilanciare le attività operative con quelle di ingegneria, guidate dallo SLO.

Il team SRE ha come obiettivo nel lungo periodo quello di **soddisfare e mantenere lo SLO**, e deve quindi difenderne il funzionamento nel breve termine. Anche a costo di utilizzare parte del suo error budget per ottenere questo bilanciamento.

Per questo ci deve essere una figura nell’organizzazione che abbia la capacità di prendere delle decisioni sui trade off tra sviluppo di funzionalità future e manutenzione del prodotto allo stato attuale. In questo senso, **l’error budget non è semplicemente un indicatore da misurare per rispettare le promesse contrattuali, ma anche un’opportunità per il team di sviluppo**, che sa quanto può investire per innovare e fin dove spingersi coi rischi.

Il rapporto tra SRE e Cloud Native

Un passaggio fondamentale della Digital Transformation, che porta i sistemi tradizionali a un approccio Cloud Native, sta nel cambiamento delle **modalità di lavoro dei team per le operazioni IT** e nelle conseguenze che ne derivano.

Il problema

Non è raro che i team non riescano a tenere il passo con la crescente richiesta di servizi. Questo succede sia nelle aziende che hanno una clientela molto ampia e in rapida crescita, sia in quelle dove la richiesta di servizi IT è alimentata dall'interno, cioè dagli altri uffici. Sono i casi in cui, paradossalmente, innovazione e successo sono concetti temuti, perché costringono i team di sviluppo a spingersi al di là delle loro stesse capacità. Come mai ciò accade?

Il problema principale è l'approccio di tipo tradizionale, che prevede **una serie molto ampia di attività manuali e ripetitive, quello che nel gergo SRE viene definito "toil", "fatica"**. L'attività manuale dei team che si occupano dell'IT rallenta tutto e rende difficile se non impossibile scaricare a terra il valore delle soluzioni esistenti, rispondendo alle richieste che si ricevono. Non parliamo poi della possibilità (che è oggi è una vera e propria necessità) di far crescere in maniera veloce applicazioni e servizi con nuove funzionalità e nuovi prodotti.

I team cercano di organizzarsi e concentrarsi su quello che ha la priorità maggiore: **fare bene il proprio lavoro, a prescindere da quello che stanno facendo gli altri team**.

Il team di sviluppo mira a portare più velocemente sempre più valore alla clientela e lo fa aggiungendo servizi e funzionalità alle applicazioni o addirittura ampliando l'offerta di prodotti dell'azienda. Invece il team delle operazioni lavora per ottimizzare il funzionamento dell'infrastruttura e delle applicazioni in maniera tale che la performance sia stabile e costante.

Questo dualismo dà vita a quello che viene chiamato "approccio a silos", in cui le attività dei vari team sono quasi completamente isolate tra di loro. Non è tanto la mancanza di comunicazione, che pure spesso c'è e ha effetti negativi, quanto il fatto di seguire processi evolutivi separati.

I team di sviluppo e di gestione delle operazioni si specializzano sempre di più **seguendo ciascuno la propria strada, dando priorità ad obiettivi diversi, organizzando autonomamente le proprie risorse**, affinando le tecnologie che ciascuno adopera senza pensare alle possibili integrazioni. Tutto ciò aggiunge complessità, diminuendo la capacità di controllo e soprattutto quella di indirizzo.

La soluzione

Per prendere delle decisioni strategiche occorre riorganizzare i team attorno a obiettivi comuni. Un compito non semplice se si considera che potrebbe essere necessario entrare in conflitto con **le scelte tecnologiche dei singoli team**, che sono state già prese da tempo e che devono essere riviste convincendo tutti i partecipanti a salire a bordo.

Tutto questo è sostanzialmente incompatibile con un approccio Cloud Native che sia anche Agile. Cosa fare, dunque? In questo contesto **la SRE costituisce un modo proattivo ed efficiente di affrontare la complessità** e i rischi dell'innovazione.

L'approccio SRE permette di superare il conflitto interno applicando le pratiche dell'ingegneria del software all'infrastruttura e alle operazioni, con l'obiettivo di creare delle piattaforme Cloud Native altamente scalabili e affidabili senza il bisogno di interventi manuali.

Il punto centrale è l'obiettivo delle aziende di diventare Cloud Native. Non bastano le nuove tecnologie per trasformare tutto: **occorre prima capire la propria architettura interna e i propri processi di manutenzione e messa in produzione del software**. E soprattutto occorre capire la propria cultura interna, che ha un ruolo cruciale per il processo di trasformazione digitale. Senza capire questi fattori prima si finisce per costruire sistemi complessi che possono essere manutenuti solo con difficoltà estrema. **Esattamente il contrario dell'approccio Cloud Native**.

Un primo passo nella direzione della SRE per applicazioni Cloud Native è quello di **creare degli standard comuni e stabili tra i differenti team**. Questo vuol dire sostanzialmente essere d'accordo su dei principi condivisi: la trasformazione infatti è possibile quando c'è una comprensione e una condivisione profonda dei principi.

Facciamo un esempio considerando **il parametro che riceve l'attenzione maggiore: la performance**. Questo elemento viene sicuramente percepito (e valutato) da tutti, all'interno e all'esterno dell'azienda. Ma la performance non è solo il funzionamento del servizio: anche i rallentamenti, i bug, le difficoltà di erogazione hanno un impatto su chi utilizza il software o il servizio. La SRE ha l'obiettivo di assicurare la disponibilità del servizio e quindi **non solo eliminare i bug ma anche identificare in maniera proattiva i potenziali problemi di performance** attraverso tutto il sistema.

Per poterlo fare occorrono delle metriche standard che servono a rendere osservabile e misurabile il sistema, e permettono così di pianificarne anche l'evoluzione.

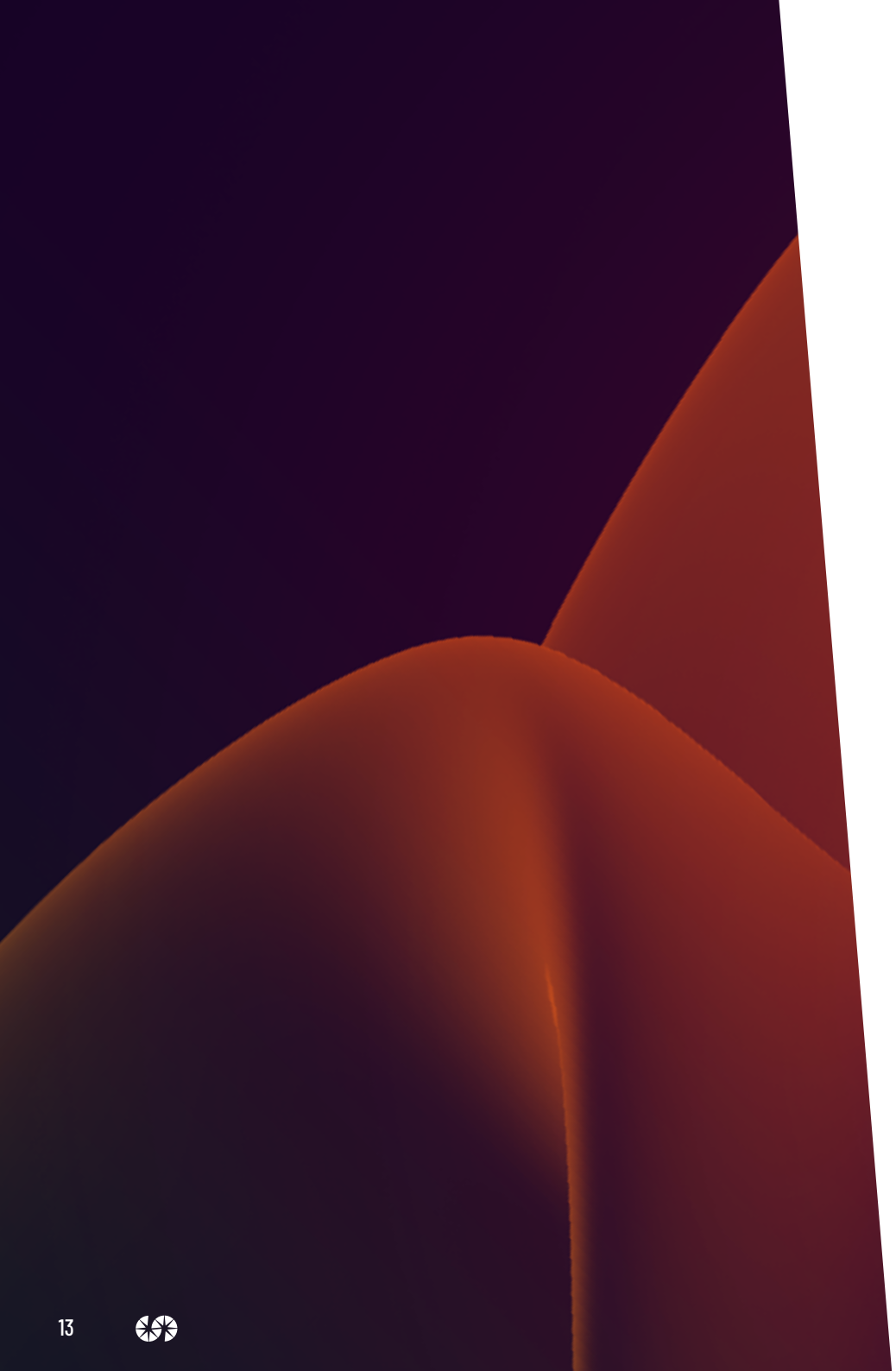
Quali metriche utilizzare? **I Four Golden Signals** funzionano sostanzialmente per la stragrande maggioranza dei servizi.
Si tratta di:

- latenza
- traffico
- errori
- saturazione

La costruzione di un'applicazione Cloud Native robusta richiede questo tipo di approccio, in cui il team SRE si occupa **non solo di collaborare alla risoluzione dei problemi del team di sviluppo** (soprattutto nelle fasi iniziali), **ma anche di mantenere la disponibilità dei servizi quando sono in produzione**.

Avendo degli SLO ben definiti e realistici, e potendo confrontarsi con un error budget che consente di prendere delle decisioni allo stesso tempo ponderate e tempestive, la piattaforma tenderà a diventare più robusta.

Diventa così possibile introdurre i cambiamenti necessari in maniera sicura e con un feedback immediato su disponibilità e performance del servizio subito dopo ogni modifica. In pratica, **si sarà messo in atto un efficace sistema di gestione del rischio**.



3 consigli per adottare la SRE

Dopo aver visto cos'è la SRE, da quale insieme di bisogni emerge e qual è il suo contesto di applicazione, passiamo all'analisi di **tre buone prassi che Google ha utilizzato nel tempo per far funzionare internamente la SRE**.

1° consiglio: sinergia tra i team

I team che si occupano dei prodotti software devono **avere a cuore l'operabilità fin dall'inizio**. Le aziende dovrebbero incentivare i team (volendo, anche economicamente) a seguire questa strada.

È buona pratica che i software engineers si occupino di verificare non solo che il codice passi i test, ma anche che vada in produzione e sia usabile correttamente.

Gli ingegneri dovrebbero iniziare a mettersi in relazione con la produzione durante le prime fasi di sviluppo: l'obiettivo di questa strategia è fare in modo che **non si creino delle fratture tra i team che si occupano dello sviluppo software e il team SRE**. Mentre i software engineers devono essere incentivati a comprendere cosa succede in produzione, il team SRE deve essere incentivato a studiare il contesto di business nel quale il software viene sviluppato, e condividerne la prospettiva.

2° consiglio: non è sempre necessario avere un team SRE separato

Spesso è possibile ottenere la maggior parte dei benefici di un approccio SRE **senza bisogno di creare un team separato dedicato alla SRE**.

Abbiamo visto come **grazie all'error budget sia possibile stabilire su cosa è necessario focalizzare l'attenzione** per aumentare l'affidabilità e l'operabilità. Affinché questo meccanismo funzioni occorre semplicemente che il team dello sviluppo software si autodisciplini e segua le regole. In particolare, si consiglia di:

- definire che **cosa è importante** per gli utenti;
- misurarlo e **definire dei "guardrail"** che si ritiene siano importanti;
- decidere **cosa fare** quando si raggiunge uno di questi guardrail;
- quando si colpisce il guardrail, **fare *davvero* ciò che si è deciso di fare**;
- **essere trasparenti** analizzando i dati e le azioni intraprese.

Se si seguono queste regole basta anche un team unico che abbia al suo interno la capacità di gestire l'error budget, senza che si debba creare un team SRE separato.

3° consiglio: non basta evitare il downtime

Mentre è compreso e accettato da tutti che lo stop del servizio colpisce direttamente gli utenti, **si deve prendere consapevolezza che anche i rallentamenti o i funzionamenti a intermittenza del servizio hanno un impatto negativo**. I team SRE devono avere un approccio più sofisticato che non semplicemente evitare il downtime. Sembra un concetto banale, ma è spesso trascurato.

L'ideale è basare il proprio approccio su un mix di valutazioni sul tipo di problemi e sul loro reale impatto in termini di affidabilità.

Ad esempio, se il 10% del traffico avviene durante le ore di picco, l'error budget si brucia più velocemente se c'è un problema in quell'arco di tempo. È in questi intervalli temporali, quindi, che occorre focalizzare l'attenzione.

Conclusione

I software moderni costruiti per funzionare su larga scala hanno bisogno di metriche di alta qualità sulla performance, che coprano una serie molto ampia di variabili e aspetti: tempo di interrogazione e risposta, latenza, capacità, variabilità, eccezioni statistiche, persistenza dei dati e molto altro. **Tutti questi elementi contribuiscono a definire l'affidabilità del software** dal punto di vista degli utenti finali e devono quindi essere misurati e capiti nelle loro varie dimensioni.

Adottare l'approccio SRE - con o senza un team ad hoc - è **il punto di partenza per evolvere le applicazioni aziendali in maniera metodica e organizzata**, partendo dai dati e scegliendo con attenzione le priorità.

Concludiamo citando le parole di **David Ferguson, EMEA lead della SRE di Google**: "Se fatta bene, la SRE è ciò che aiuta l'organizzazione a mantenere le promesse fatte sulla bontà dei propri prodotti".


SPARKFABRIK