JAVA & COLD START?

in 5 rapidi utili consigli…

GIANNI FORLASTRO

CTO @ finwave

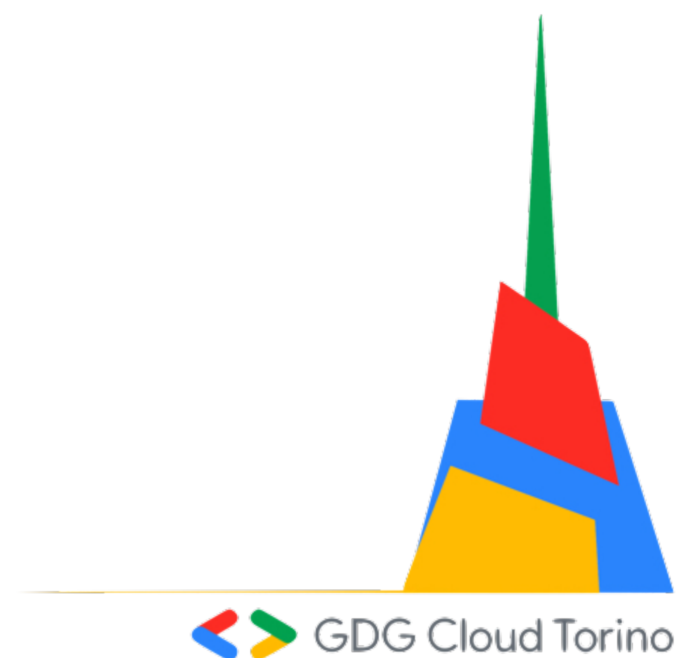
Google Cloud Champion Innovator

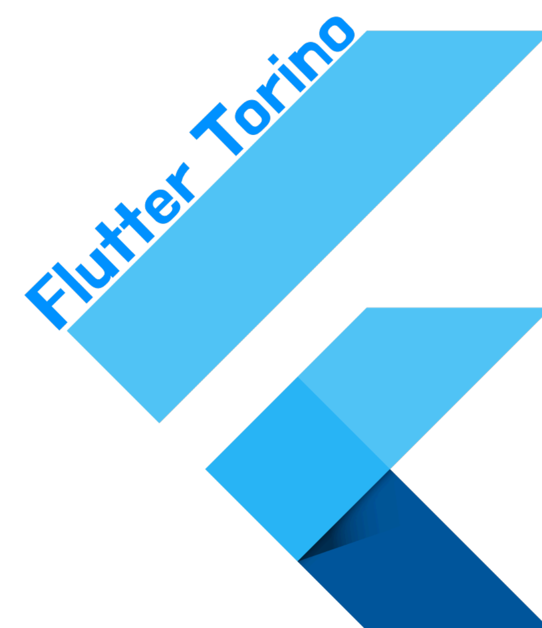
Community Lead
@ GDG Cloud Torino
@ Flutter Torino

FRANCESCO PIRRONE

Full stack Dev @ finwave

Community Lead
@ GDG Cloud Torino
@ Flutter Torino

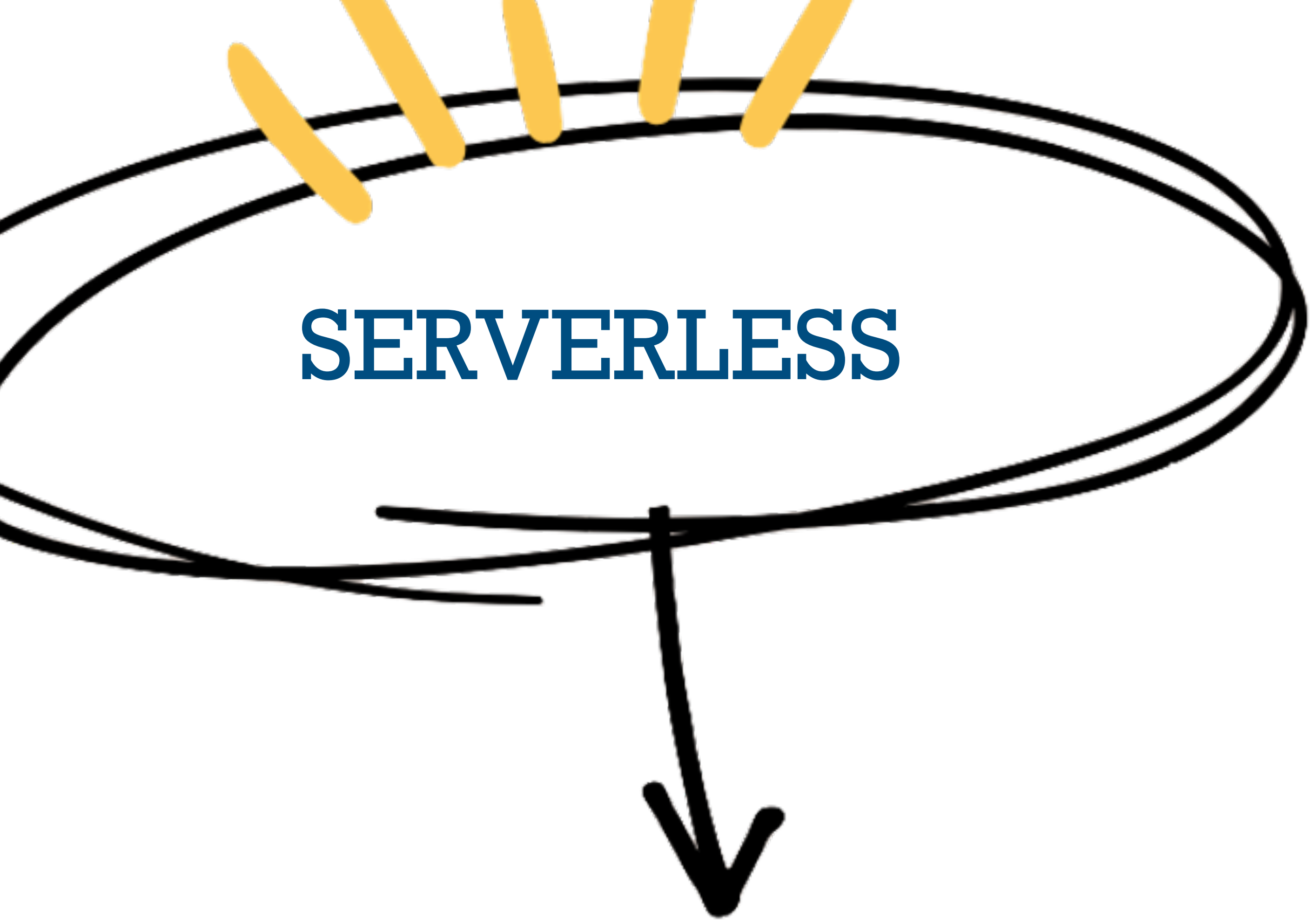




- Performance (avvio,throughtput)
- Efficienza (utilizzo/consumo risorse)
- Costi / Benefici (Cost-effective)
- Scalabilità e resilienza
- Sostenibilità e manutenibilità

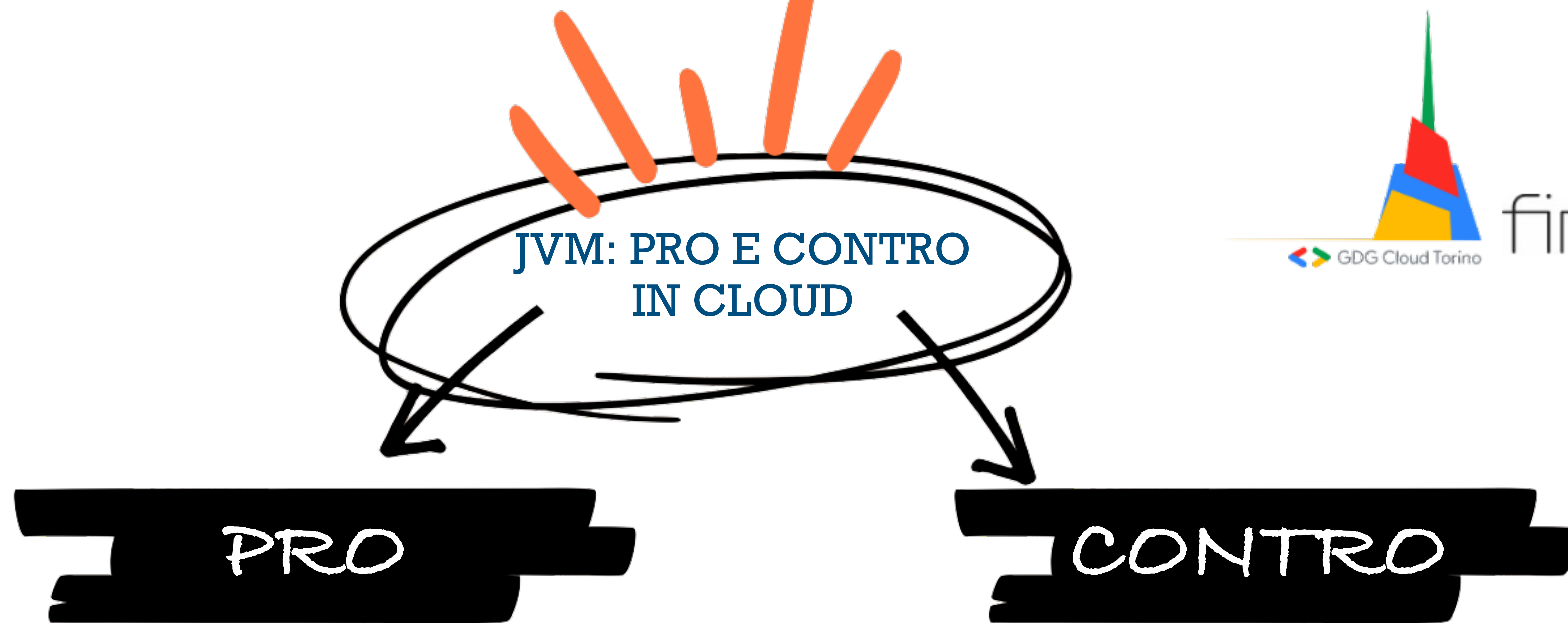
Nello sviluppo software le performance è l'unico fattore di scelta.

Molto spesso i tempi di sviluppo e manutenzione hanno un impatto maggiore: riscrivere tutto in RUST non è una strategia percorribile.



Da una architettura Serverless, ci si aspetta:

- Semplice nella gestione: ambiente di runtime gestito da terzi
- Efficiente nei costi: costi basati sull'utilizzo effettivo, nessun costo in caso di non utilizzo
- Veloce
- Flessibile: autoscaling, adattabile, …
- Semplice nello sviluppo: ci si deve concentrare sul prodotto e non sull'infrastruttura



GDG Cloud Torino finwave

- Ecosistema robusto
- Performante*
- Strumenti di sviluppo maturo
- Largo supporto
- In continua evoluzione
- Sicuro
- Ampiamente noto e utilizzato

- Tempo di avvio (cold start)
- Utilizzo elevato di memoria

GDG Cloud Torino

finwave

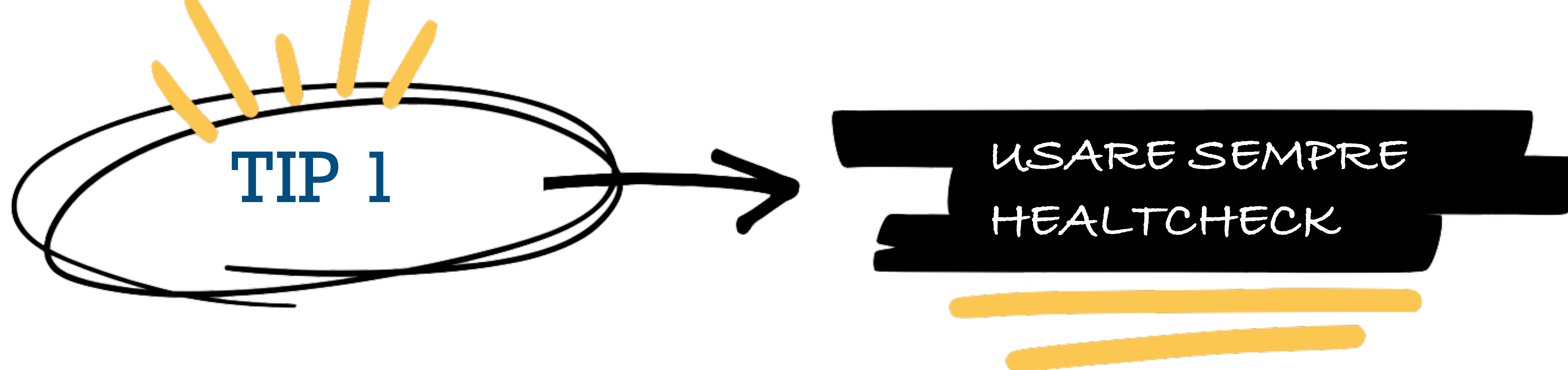
Sfruttare meglio l'autoscaling: prima sono disponibili nuove repliche, meno si rischia un sovraccarico.

Un utilizzo granulare dell'autoscaling implica anche costi minori evitando l'overprovisioning

Maggiore resilienza: in caso di disservizi infrastrutturali è facilmente ripristinabile l'operatività e ridurne l'impatto.

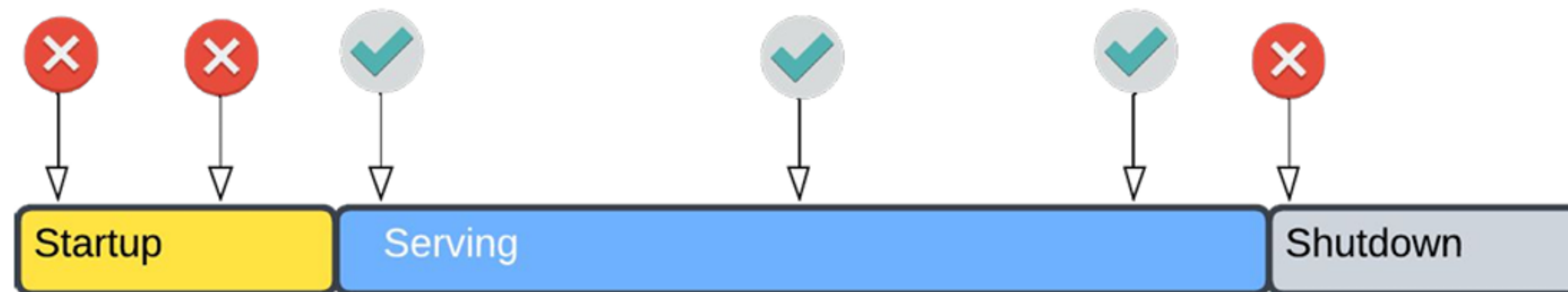
Anche gli aggiornamenti del software risultano più rapidi

1. USARE SEMPRE HEALTCHECK
2. CPU BOOST
3. CAMBIARE RUNTIME
4. UTILIZZARE COMPILAZIONE AOT
5. UTILIZZARE CRaC



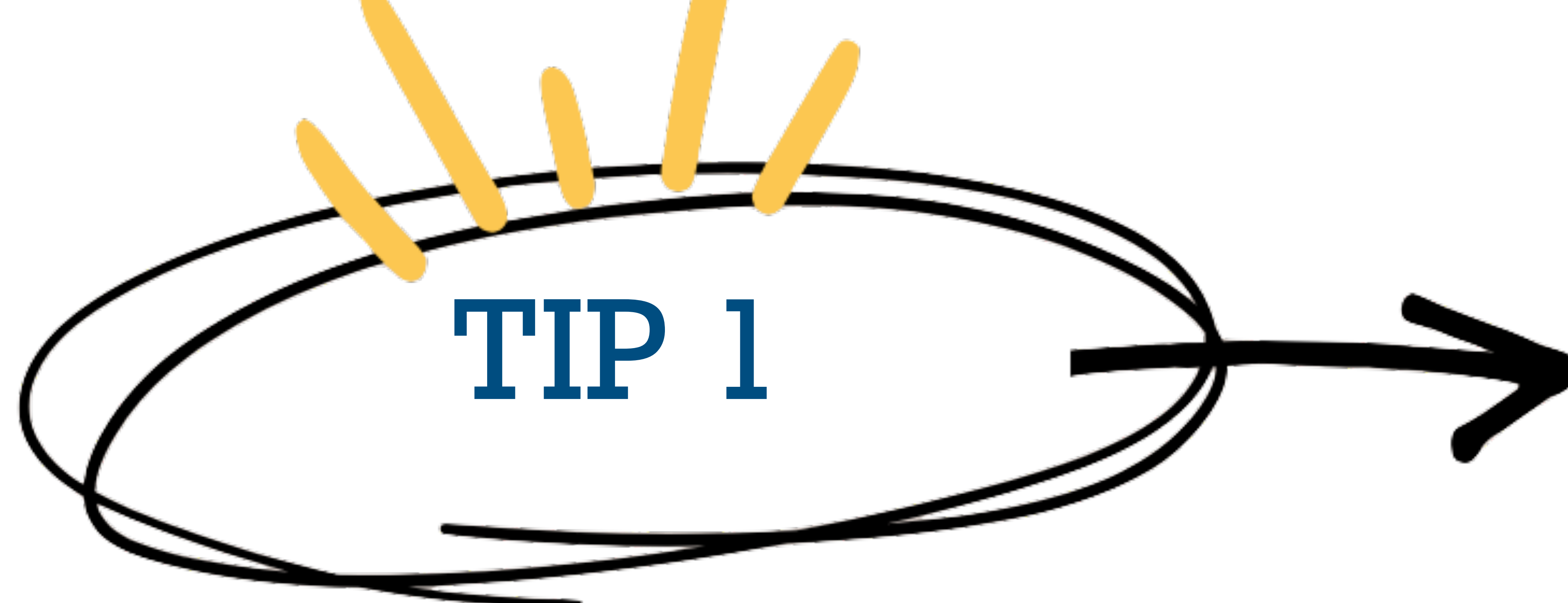
Impostare un healtcheck di avvio (readiness/startup) impedisce che un servizio riceva traffico prima che sia pronto a gestirne le richieste (leggasi application context di Spring).

Un healtcheck che verifichi la disponibilità del servizio (liveness) impedisce che questo riceva traffico in fase di shutdown o di guasti (long garbage collector).
Inoltre molte piattaforme collegano alla liveness il self-healing (riavvio automatico).



GDG Cloud Torino

finwave



USARE SEMPRE HEALTCHECK

Cloud Run ← Esegui il deployment della revisione su java-serverless-spring-boost (us-central1)

Nome container: Undefined parameter - autogenerateName EDIT

Comando container

Lascia vuoto per utilizzare il comando del punto di ingresso definito nell'immagine container.

Argomenti container

Argomenti passati al comando del punto di accesso.

Risorse

Memoria: 512 MiB — Memoria da allocare a ogni istanza di questo container.

CPU: 1 — Numero di vCPU allocate a ogni istanza di questo container.

Controlli di integrità

Startup probe	http /actuator/health/readiness every 240s
Ritardo iniziale	0s
Timeout	240s
Soglia di errore	1

Liveness probe	http /actuator/health/liveness every 10s
Ritardo iniziale	0s
Timeout	1s
Soglia di errore	3

+ AGGIUNGI CONTROLLO DI INTEGRITÀ

FINE

AGGIUNGI CONTAINER

Seleziona il tipo di controllo di integrità: Controllo dell'attività

Seleziona il tipo di probe *: HTTP

Percorso (ad. es. /ready) *: /actuator/health/liveness — Percorso di accesso al server HTTP.

Porta *: 8080 — Porta per stabilire una connessione per l'istanza Cloud Run. Verrà utilizzata per impostazione predefinita la porta del container, il cui valore predefinito è 8080.

Intestazioni HTTP (facoltativo) — Intestazioni personalizzate da impostare nella richiesta. HTTP consente intestazioni ripetute.

Ritardo iniziale *: 0 secondi — Numero di secondi prima dell'esecuzione del primo probe dopo l'avvio del container. Inserisci un valore intero compreso tra 0 e 3600.

Periodo *: 10 secondi — Numero di secondi di attesa tra l'avvio di un tentativo di probe. Inserisci un valore intero compreso tra 1 e 3600.

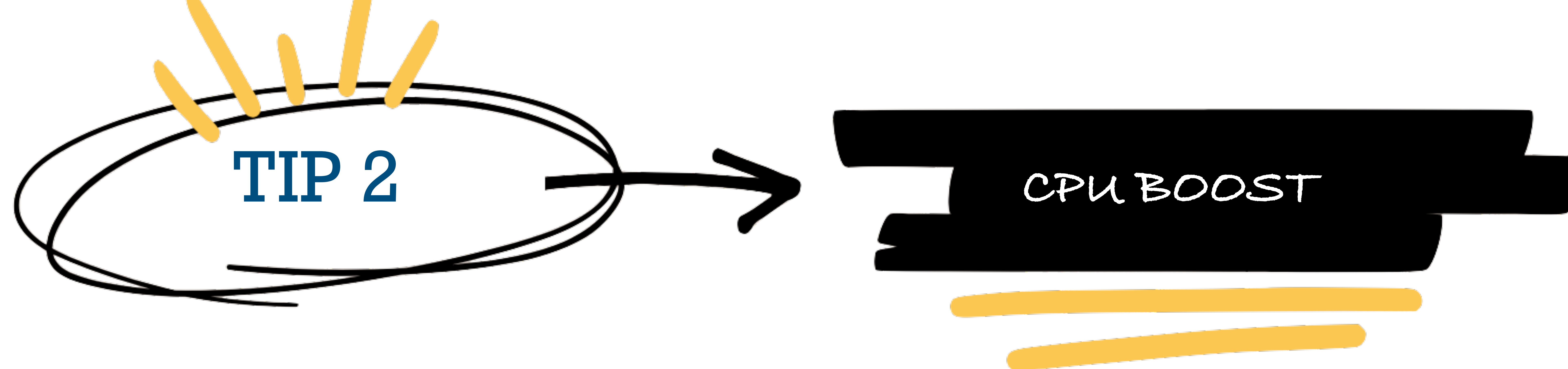
Soglia di errore *: 3 — Numero di tentativi del probe prima di contrassegnare il container come Non pronto.

Timeout *: 1 secondi — Numero di secondi che devono trascorrere per il time out del probe. Questo valore non deve superare i secondi del periodo. Inserisci un valore intero compreso tra 1 e 3600.

AGGIORNA ANNULLA

Su **Cloud RUN** (e KNATIVE) è possibile impostare i controlli di attività e avvio

GDG Cloud Torino finwave

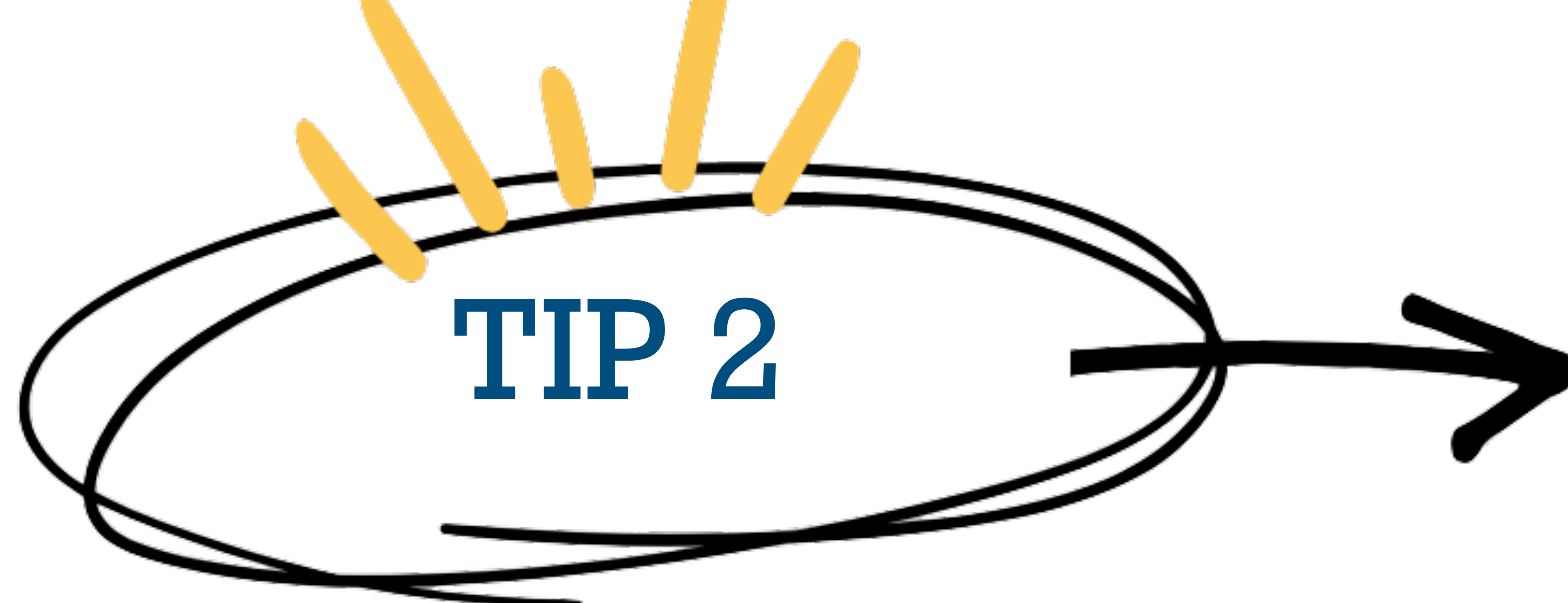


La fase di startup della JVM richiede sempre maggiore computazione rispetto all'esecuzione.

La JVM deve compilare le classi per l'architettura hardware e i framework devono inizializzare il context applicativo. Per questo motivo avere ***una potenza extra di CPU*** a disposizione all'avvio riduce i tempi di start.

Su Kubernetes è possibile impostare un *CPU Limits* > *CPU Requests*.

Attenzione: la CPU Limits non è garantita che sia sempre a disposizione.



- (●) La CPU viene allocata solo durante l'elaborazione delle richieste
  Ti vengono addebitati i costi per richiesta e solo quando l'istanza di container elabora una richiesta.
- ( ) La CPU è sempre allocata
  Ti verranno addebitati i costi per l'intero ciclo di vita dell'istanza del container.

Ambiente di esecuzione

L'ambiente di esecuzione in cui viene eseguito il container. Scopri di più

- ( ) Predefinita
  Cloud Run seleziona automaticamente l'ambiente di esecuzione adatto.
- ( ) Prima generazione
  Avvii completi più rapidi.
- (●) Seconda generazione
  Supporto del file system di rete, compatibilità Linux completa, prestazioni della CPU e di rete più rapide.

Scalabilità automatica revisione

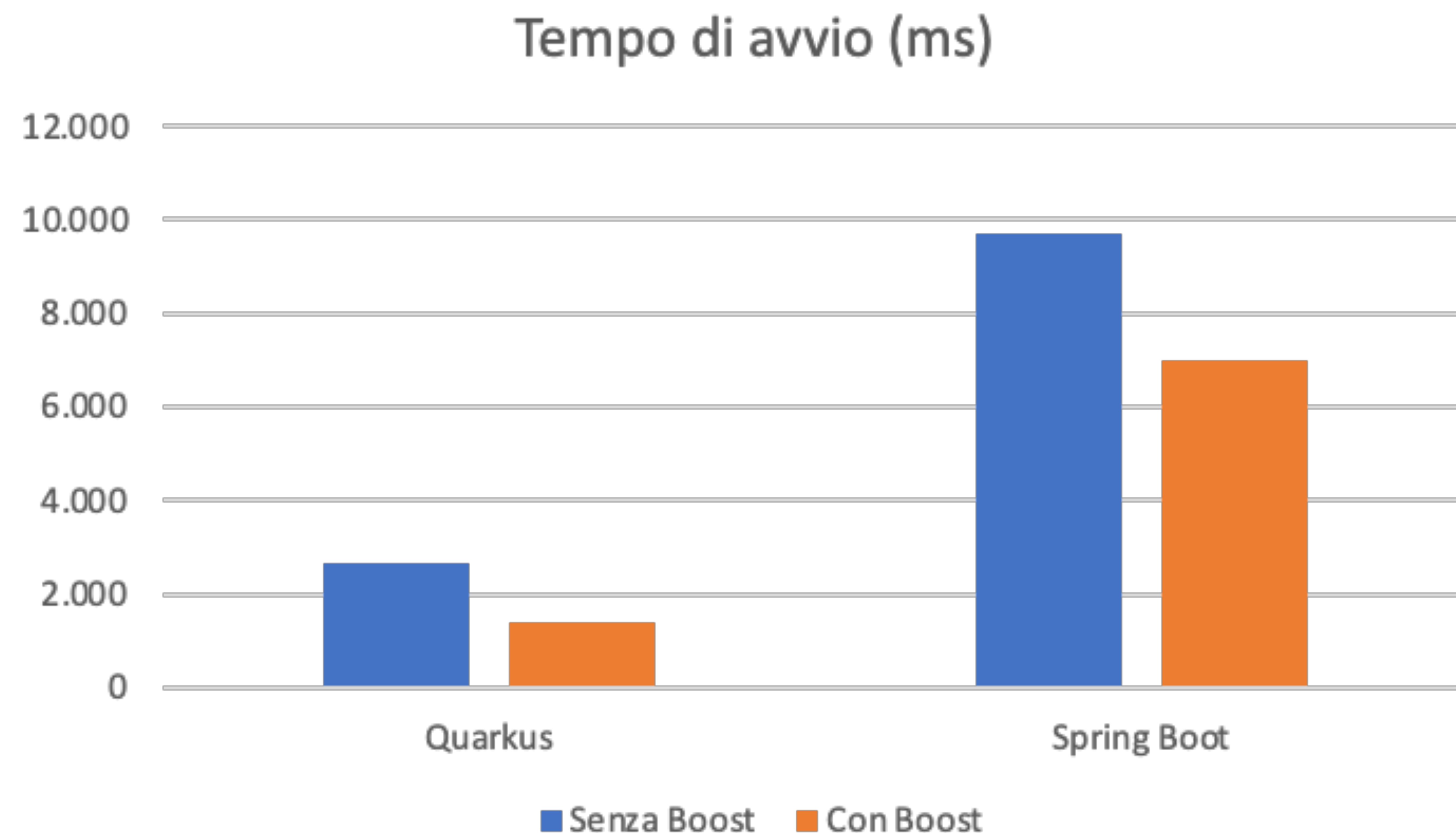
Numero minimo e massimo di istanze per la nuova revisione.

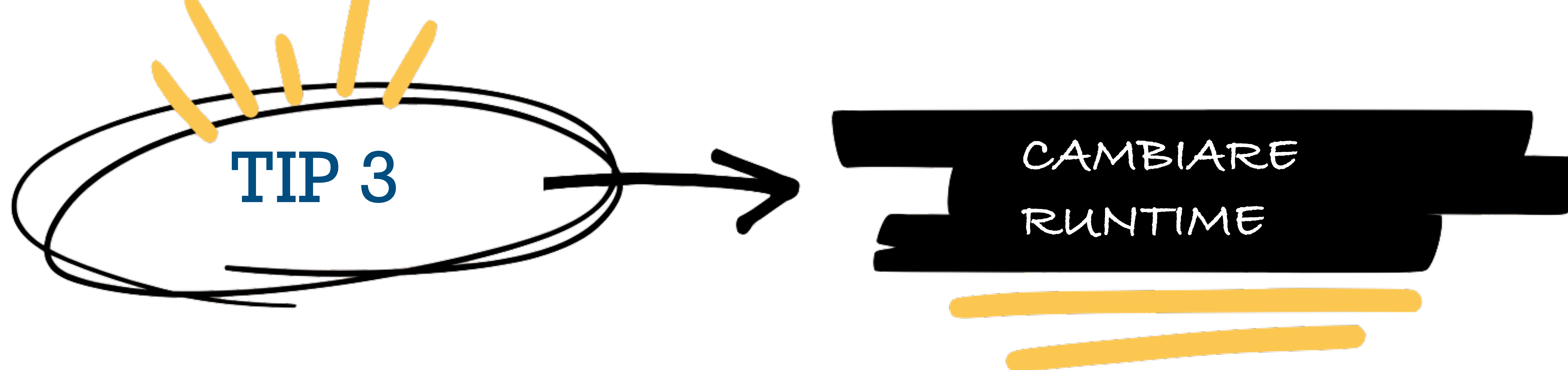
Numero minimo di istanze *: 0

Numero massimo di istanze *: 100

Nella maggior parte dei casi d'uso è preferibile il numero minimo di istanze. Usa questa impostazione solo se hai bisogno di impostazioni specifiche per ogni revisione.

- [x] Boosting della CPU all'avvio
  Avvia i container più velocemente allocando più CPU durante il tempo di avvio. Scopri di più



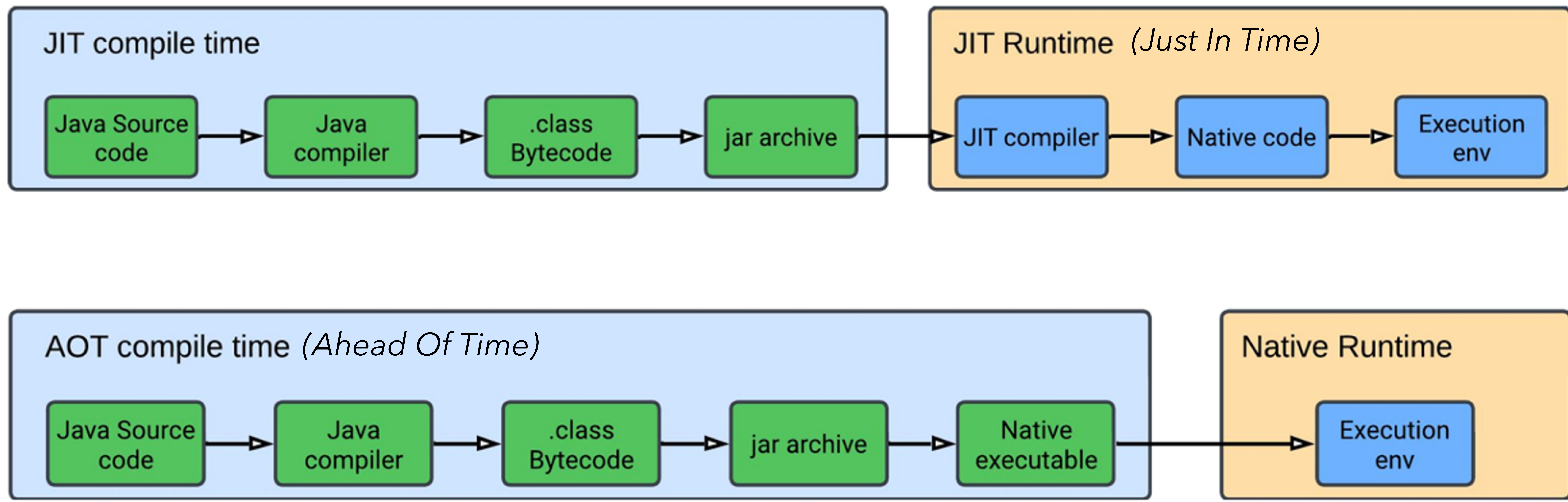


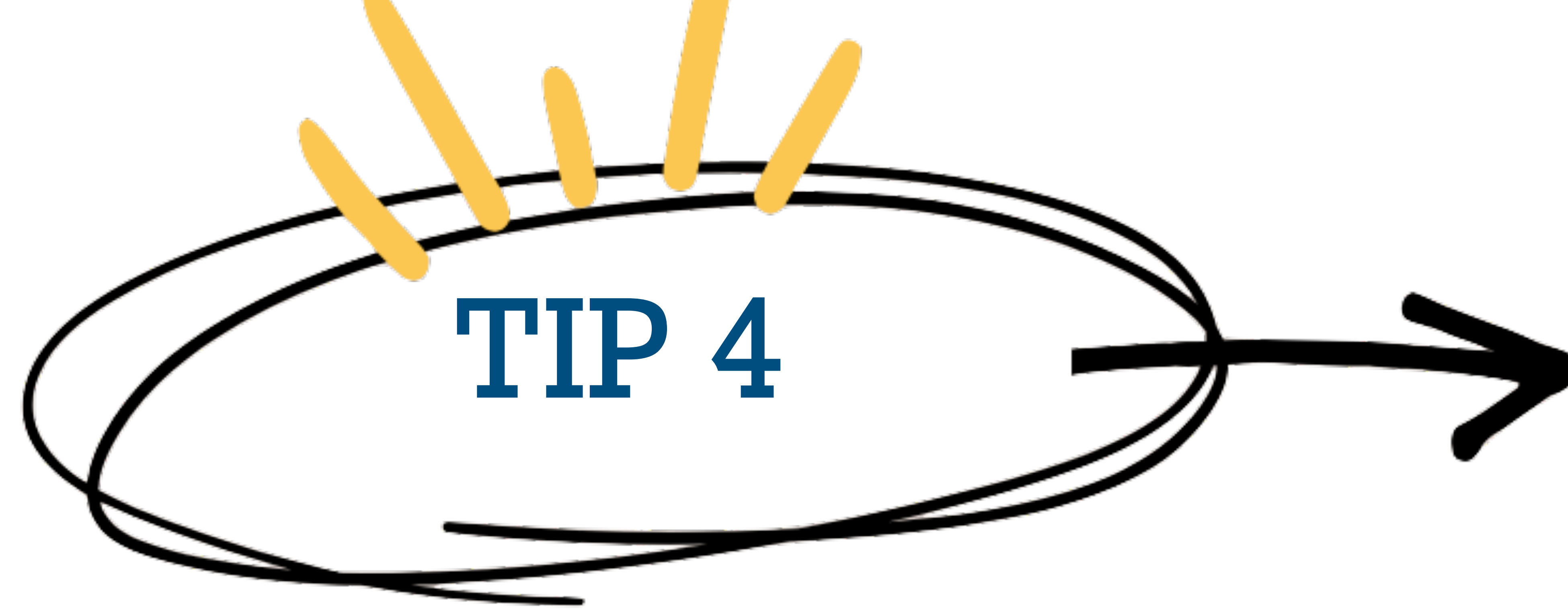
- Ad ogni versione della JDK si stanno introducendo ottimizzazioni soprattutto in ambito container/tempo di avvio.
- Usare framework più compatti come Quarkus o Micronaut riduce l'uso di package scan e reflections in fase di avvio
- Ridurre la dimensione dell'immagine dei container, ad esempio usare alpine ed usare la JRE anziché una immagine jdk-devel. Minore è la dimensione dell'immagine, più veloce risulta la fase di pull del container.
- Preferire l'avvio tramite classpath/layer anziché tramite Fat-JAR

GDG Cloud Torino finwave

TIP 4

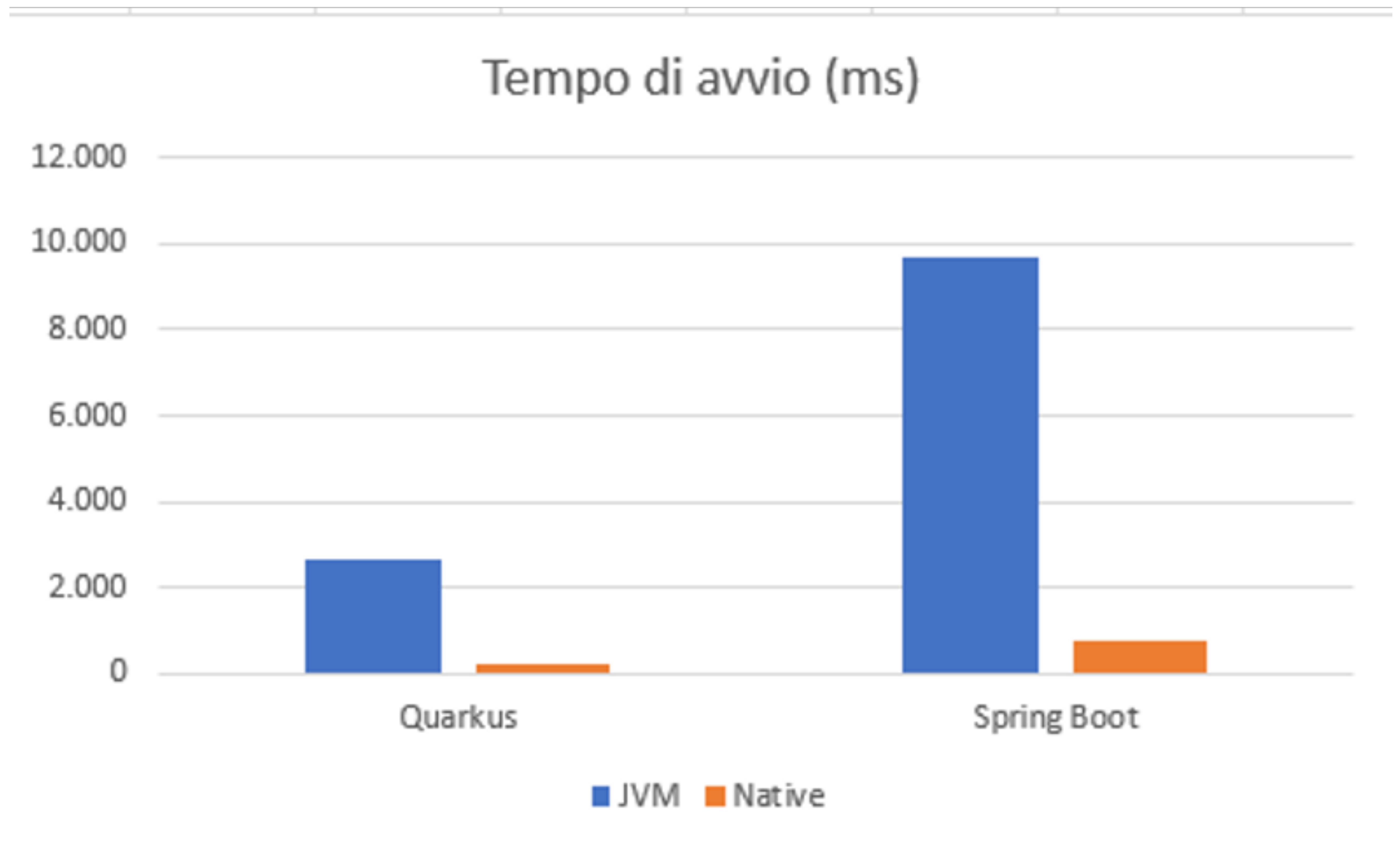
UTILIZZARE COMPILAZIONE AOT

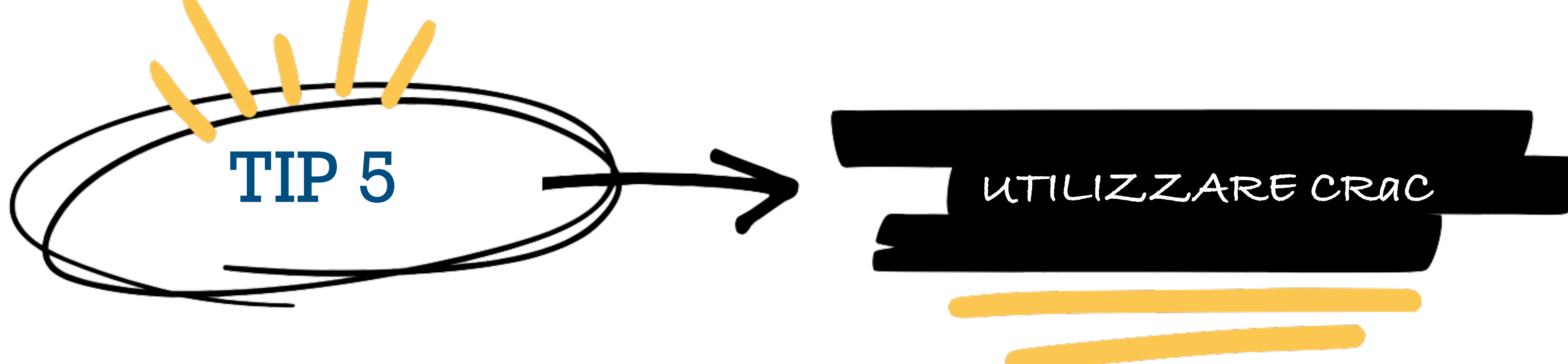




UTILIZZARE COMPILAZIONE AOT

- Molti framework supportano la compilazione nativa in GraalVM.
- Sono necessari degli hints per supportare la reflections (largamente usata da librerie e framework)
- Tuttavia questo porta ad una differenza di comportamento tra il runtime in locale e in esecuzione.
- Tempi di compilazione aumentati





CRaC : Coordinated Restore at Checkpoint

JIT compile time

Java Source code → Java compiler → .class Bytecode → jar archive

JIT Runtime

JIT compiler → Native code → Execution env

JIT compile time

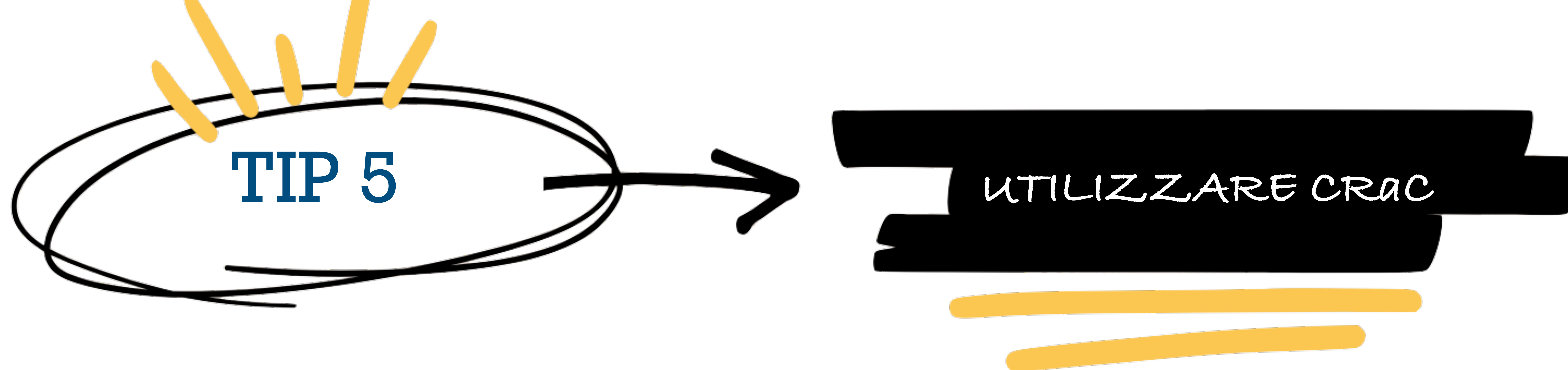
Java Source code → Java compiler → .class Bytecode → jar archive

JIT Runtime

App run → Snapshot app in Disk → Execution env

Checkpoint

Restore

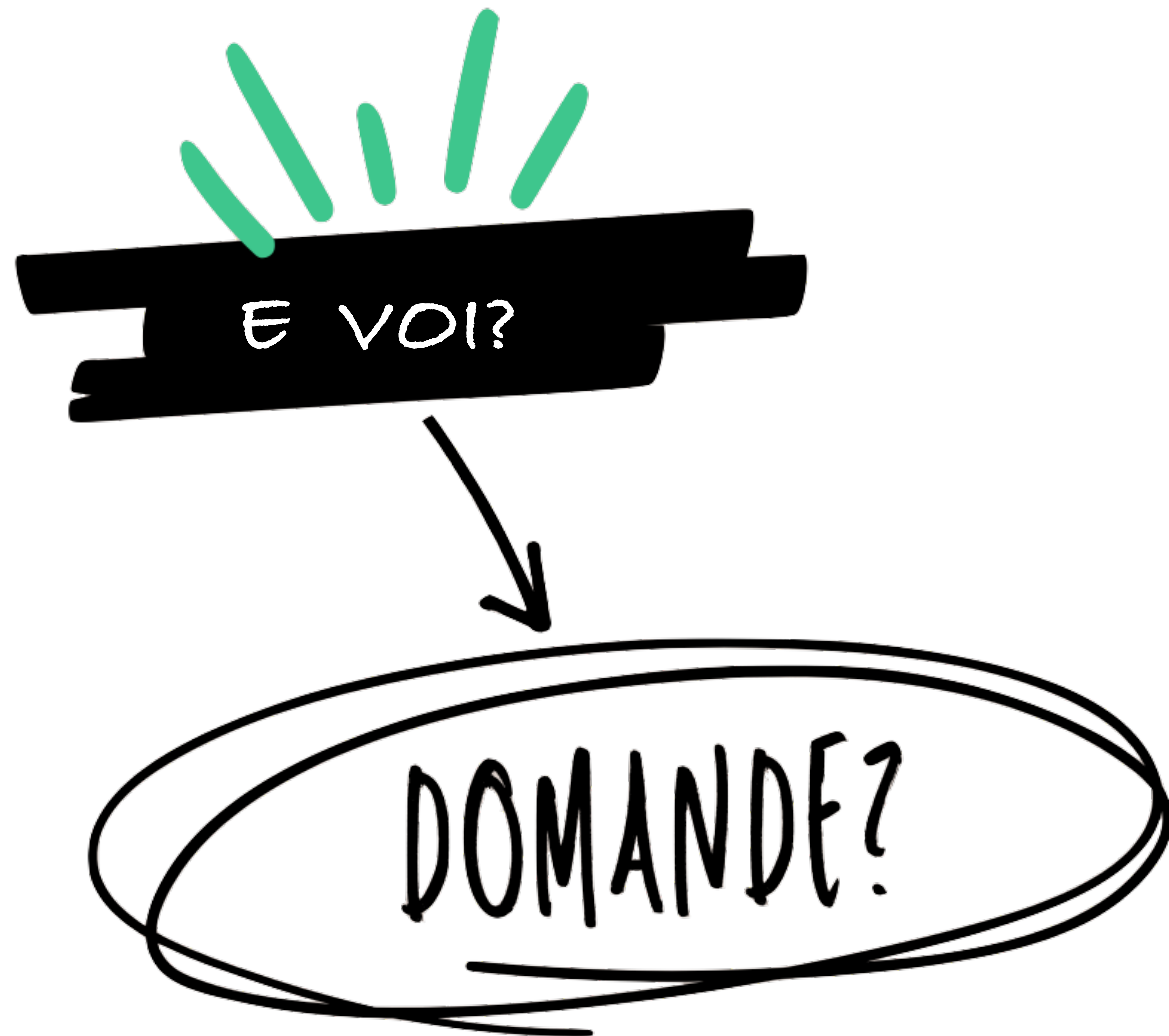


- Alternativa alla compilazione AOT
- Incrementa i tempi di avvio
- Tuttavia è necessario creare un punto di ripristino per l'applicazione.
  Nel punto di ripristino sono conservate anche variabili d'ambiente e secret.
  In caso di modifica di queste è necessario creare un nuovo punto di ripristino cancellando il precedente ed avviare l'applicativo a freddo.
- Usa un dump della memoria per ripristinare l'applicativo, questo comporta saltare la fase di esecuzione
- Disponibile anche su AWS Lambda tramite SnapStart

GDG Cloud Torino
finwave

CloudConf 2024

DOMANDE?



- Avete mai usato AOT o CraC?
- Avete avuto problemi con Cold Start?
- Avete mai applicato qualcosa di questo tipo su Cloud Run o KNATIVE?
- Quali sono le vostre tecniche per ottimizzare i tempi di avvio?

GDG Cloud Torino finwave

JAI CAMPBELL

Cloud Architect | SRE | Google Cloud Specialist

"EXPLORE SITE RELIABILITY ENGINEERING BEST PRACTICES WITH GEMINI".

NICOLA GUGLIELMI

Google Cloud Architect | Google Cloud Authorized Trainer

"EVOLUZIONE DELLE ARCHITETTURE CLOUD SCALANDO DA 100 A 100M UTENTI!"

APPUNTAMENTO ONLINE, ISCRIZIONE GRATUITA

18 GIUGNO 2024

ORE 17.00

FREE TICKETS:

GDG Cloud Torino