





Monitoring di microservizi con Prometheus

Webinar



Marco Breveglieri

-  Software Developer
-  Trainer and Consultant
-  Tech Content Creator
-  Community lover

 <https://www.breveglieri.it>



Premessa



Servizi, micro e non...

- I **microservizi** sono senz'altro una delle architetture moderne più efficaci
 - Veloci, scalabili, flessibili, isolabili, sicuri, «cloud-ready»...
 - Complessi da ideare, progettare e implementare (se «veraci»)
- **Ciò che vedremo si può applicare tranquillamente a...**
 - Applicazioni e/o servizi che girano in background (es. Windows Services)
 - Applicazioni Web (es. ASP.NET Web API, ASP.NET MVC, ...)
 - Processi «long running» che eseguono una certa «business logic»



L'importanza del «controllo»



**«La potenza
è nulla
senza
controllo»**

(Rocco Siffredi)

...in che modo?



WHO'S DEAD??

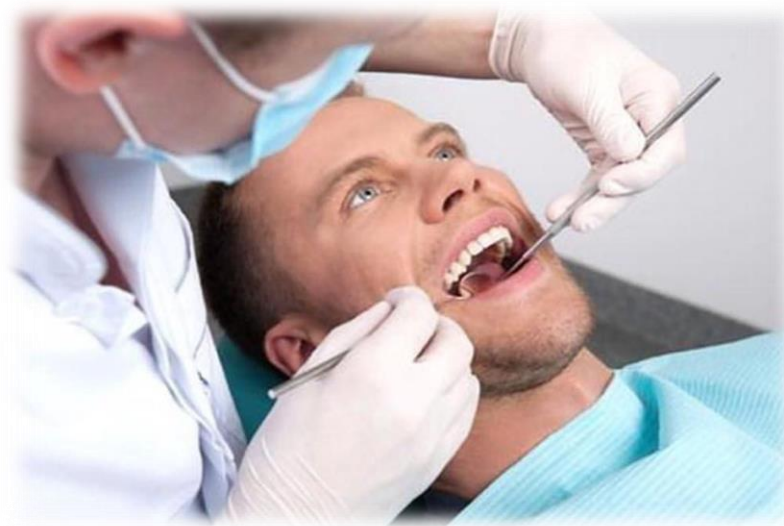
- **Raccolta e archiviazione di log**
 - Strumento essenziale di tracciamento
 - Facile da implementare
 - Contengono prevalentemente informazioni rilevanti per il business
 - Difficili da usare per monitoraggio in tempo reale: dati voluminosi e non aggregati
 - Non adatti a interventi preventivi
- **Tool generici (Task Manager, Event Monitor, Query WMI)**
 - Troppo blandi, generici, non integrati
 - Difficilmente automatizzabili

Ci vorrebbe...

...un «pattugliamento» costante e affidabile del sistema software!

- Lettura continua di variabili interessanti
- Focus su elementi di livello alto e basso
- Verifiche estese a tutti i tipi di servizi
- Acquisizione di grandezze eterogenee
- Archiviazione dei dati in modo massivo
- Possibilità di consultazione agevolata
- Indipendenza hardware del monitoring

...perché «prevenire è meglio che curare».



Conosciamo
Prometheus!



Che cos'è Prometheus?

- **Tool di monitoraggio e alerting di sistemi software**

- Nato originariamente in *SoundCloud*
- Prodotto opensource, adottato da molte grandi aziende
- Progetto indipendente e completamente autonomo
- Community di sviluppatori e utenti molto attiva
- Realizzato in linguaggio Go e crossplatform

Sito ufficiale (con risorse e documentazione)

👉 <https://prometheus.io/>



Che cosa fa?

- **Raccoglie e archivia dati con un riferimento temporale (TSDB)**
 - Contatta i servizi (es. tramite endpoint configurati) verificandone lo stato
 - I dati vengono salvati unitamente a un identificativo e a un «timestamp»
 - E' possibile memorizzare dati relativi a grandezze eterogenee
 - Le grandezze possono essere associate a «etichette» personalizzate
 - Le grandezze prendono il nome di «metriche»
- **Offre una dashboard (minimale) per visualizzare i dati archiviati**
 - E' possibile visualizzare lo stato dei compiti configurati
 - E' possibile eseguire semplici query per graficare i dati

Caratteristiche principali

- **Modello dati multidimensionale** con valori organizzati per metriche aventi un nome (ed eventuali etichette con il relativo valore associato)
- **Nessuna dipendenza da altri storage** (es. database server): ogni nodo è autonomo
- Supporto a *PromQL*, un semplice linguaggio integrato per **query sui dati**
- Il recupero dati dai target **in modalità «pull»** ne semplifica l'adozione
- **Si integra facilmente con altri tool** per l'esposizione dati a scopo di visualizzazione e/o elaborazione successiva degli stessi (es. *Grafana*)

Acquisizione dei dati (*scraping*)

- **La modalità «pull» di recupero dei dati offre alcuni vantaggi**
 - Semplifica l'attivazione del monitoring anche su una macchina di sviluppo
 - Consente di determinare automaticamente se un certo nodo è «down»
 - Verificare stato e veridicità dei dati è possibile accedendo semplicemente all'endpoint
- **Grazie ai «Push Gateway», è possibile anche per quei processi che eseguono per un tempo (molto) limitato**

Demo Time! 🕒

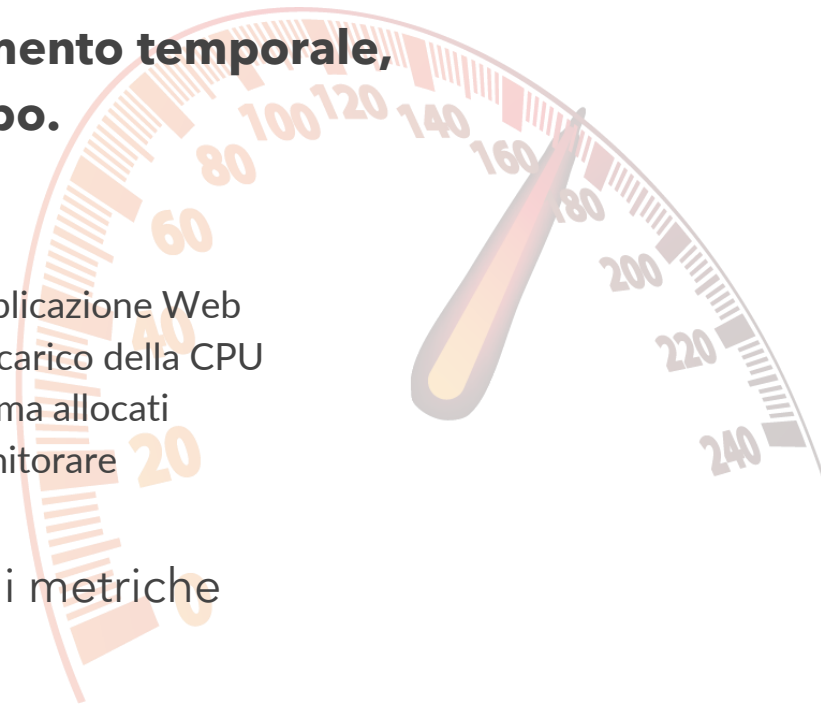
- Scarichiamo e installiamo Prometheus
- Esploriamo il file di configurazione iniziale e personalizziamolo
- Muoviamo i primi passi nella dashboard
- Diamo una prima sbirciata alle metriche



Le metriche

Misurazioni rilevate, associate a un riferimento temporale, e acquisite in modo continuativo nel tempo.

- Alcuni esempi di metriche:
 - Numero di richieste concorrenti o totali in un'applicazione Web
 - Memoria occupata dal processo in esecuzione e carico della CPU
 - Numero di handle, thread e altri oggetti del sistema allocati
 - Tutto ciò che possiamo ritenere rilevante da monitorare
- *Prometheus* supporta diverse tipologie di metriche



Tipologie di metriche

- **Counter**

- Rappresenta un contatore sempre crescente (il valore può solo aumentare)
- Una volta azzerato all'avvio, può essere incrementato di una o più unità

Esempi: numero di attività completate, numero di richieste ricevute, numero errori.



- **Gauge**

- Rappresenta un valore numerico che può arbitrariamente salire o scendere
- Può essere incrementato (o decrementato) o impostato a un valore specifico

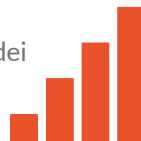
Esempi: percentuale di CPU utilizzata, memoria allocata, temperatura rilevata



- **Histogram (*)**

- Campiona valori di una grandezza osservata suddividendoli in range (configurabili)
- Calcola e fornisce conteggio, media e percentili di tutti i valori raccolti

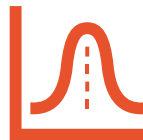
Esempi: durata di richieste HTTP, dimensioni dei pacchetti in ingresso



- **Summary (*)**

- Simile alla metrica «Histogram» ma suddivide i valori in bucket statistici (quartili)
- Si usa al posto di «Histogram» quando non è possibile stabilire range a priori

Esempi: durata di richieste HTTP, dimensioni dei pacchetti in ingresso



Esposizione delle metriche

- **Le «client libraries» sono librerie scritte in diversi linguaggi di programmazione per esportare le metriche da applicazioni e servizi**
 - Semplificano la strumentazione del proprio codice
 - Consentono la raccolta dei dati per produrre metriche
 - Espongono a Prometheus i dati raccolti nel formato corretto
 - Si integrano con l'architettura per esporre endpoint delle metriche (*)
 - Offrono supporto per la creazione di collezioni di dati custom (*)

() si tratta di servizi opzionali*

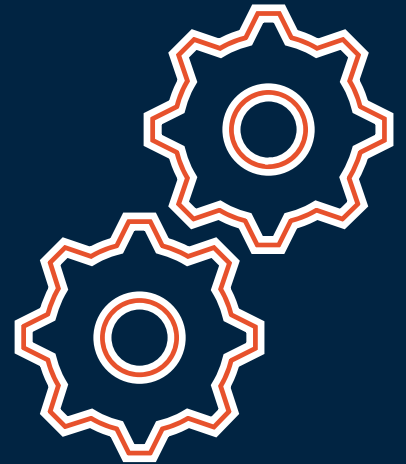
- **Sono disponibili per la maggior parte dei linguaggi più diffusi**
 - Go, Java, Python, Ruby (ufficiali)
 - C/C++, Dart, Delphi, Elixir, Haskell, LUA, .NET/C#, Node.JS, Perl, PHP, R (non ufficiali)

Demo Time! 🕒

- Creiamo una basilare applicazione Web
- Configuriamo un nuovo job su *Prometheus*
- Esponiamo alcune metriche di esempio
- Ispezioniamo lo stato del job e le metriche raccolte con *PromQL*



C'è di più...



Exporters

- Grazie agli exporters, Prometheus può raccogliere dati da altri sistemi
 - Sono disponibili servizi per esportare metriche verso Prometheus
- E' possibile acquisire informazioni sul singolo «nodo»
 - Sono disponibili exporter per la maggior parte dei sistemi operativi
 - Grazie a questi exporter, si possono collezionare metriche relative alla macchina, allo stato dell'hardware, al sistema operativo e ad alcuni servizi di base
- Esistono «exporter» anche per i sistemi di messaggistica più diffusi
 - NATS e RabbitMQ consentono l'esportazione di metriche nel formato richiesto
 - I sistemi di messaggistica possono essere monitorati assieme ai propri servizi

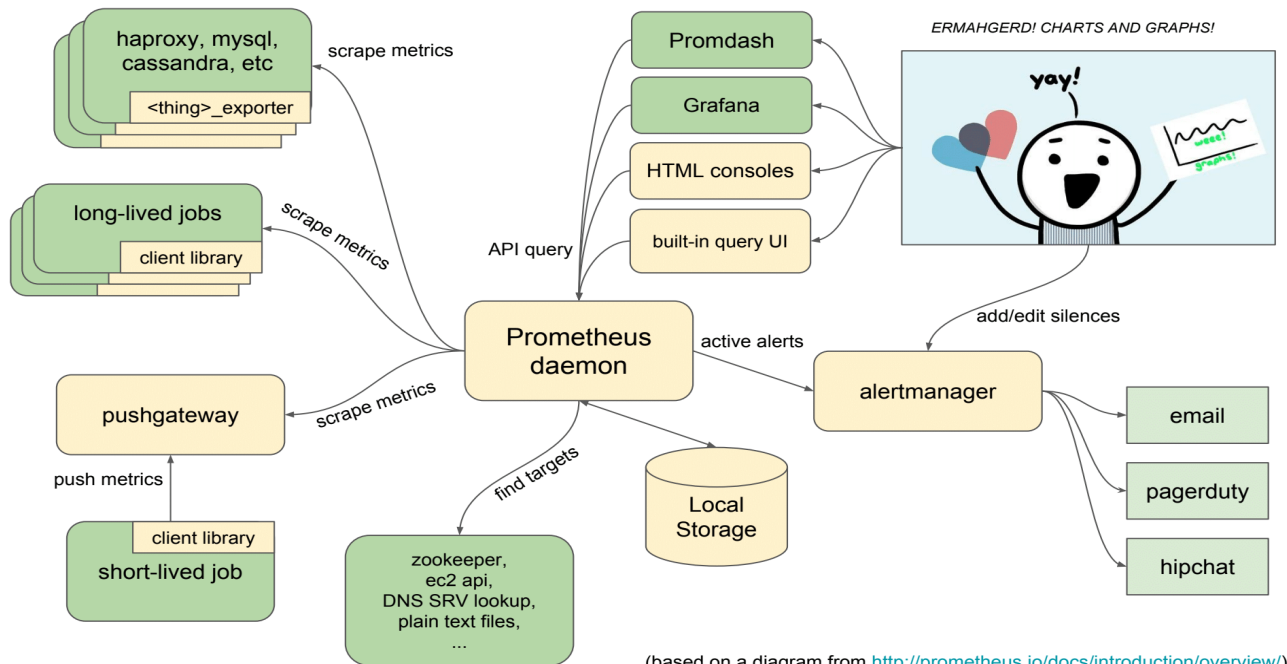
Alerting

- Prometheus supporta un **sistema di «alerting» integrato**
 - E' possibile definire regole che determinano una situazione di «all'erta» da mostrare su dashboard
- ***Alertmanager***: tool per raccogliere e inviare alert
 - Consente di ricevere gli alert e di mostrarli separatamente
 - Invia avvisi relativi agli alert sfruttando i canali più diffusi
 - Invio mail tramite SMTP
 - HTTP POST verso endpoint configurabile
 - Invio di un messaggio su bot Telegram (!)

...e molto altro!

- Prometheus supporta **interfacce configurabili** per storage alternativi
 - E' possibile sostituire lo storage locale (consigliato) con altre basi di dati
- Prometheus è **disponibile anche per Docker**
 - Il supporto Docker rende possibile «pacchettizzare» tool, app e configurazioni
 - Una volta creata l'architettura, questa può essere esportata facilmente nel cloud

Un tipico scenario



(based on a diagram from <http://prometheus.io/docs/introduction/overview/>)

Alcune limitazioni...

- **Prometheus *non* è progettato per «long time storage»**
 - I dati delle metriche vengono conservati per un tempo relativamente breve
- **Prometheus *non* è un tool di «data mining»**
 - La semplicità di *PromQL* è un indizio del fatto che il tool non nasce per analisi, elaborazione ed estrazione di informazioni dai dati raccolti
- **Prometheus *non* è un sistema di business intelligent reporting**
 - Per necessità di alto livello o visualizzazioni intelligibili, servono altri tool (es. *Grafana*)

Q & A

Domande e (forse) risposte



Grazie!

