Tieni sotto controllo

i tuoi (micro)servizi Delphi con Prometheus e Grafana

MARCO BREVEGLIERI

ABLS Team SNC

Software and Web Developer, Trainer and Consultant

DelphiDay italian conference

ONLINE EDITION

wintech italia

Dove seguirmi

https://www.breveglieri.it

https://www.twitch.tv/compilaquindiva

https://bit.ly/yt-compilaquindiva

https://www.compilaquindiva.com/

https://www.delphipodcast.com/

https://www.linkedin.com/in/marcobreveglieri/

https://twitter.com/mbreveglieri

https://github.com/marcobreveglieri

https://www.facebook.com/marcobreveglieri

DelphiDay italian conference

ONLINE EDITION

wintech italia

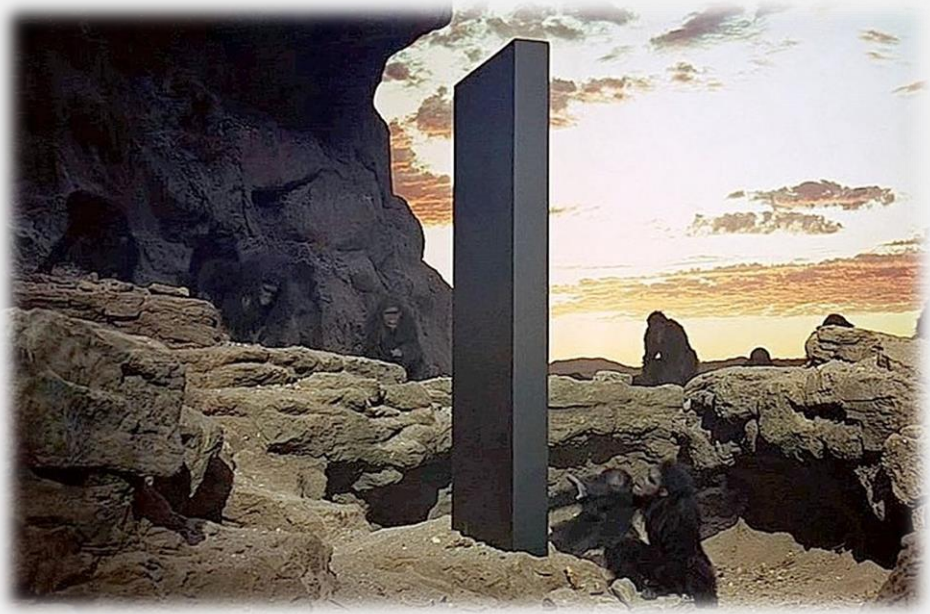
Prima di iniziare...

Non solo «(micro)servizi»

- I «microservice» rappresentano senz'altro una delle architetture moderne più efficaci
  - Veloci, scalabili, flessibili, isolabili, sicuri, «cloud-ready»...
  - Complessi da ideare, progettare e implementare (se «veraci»)

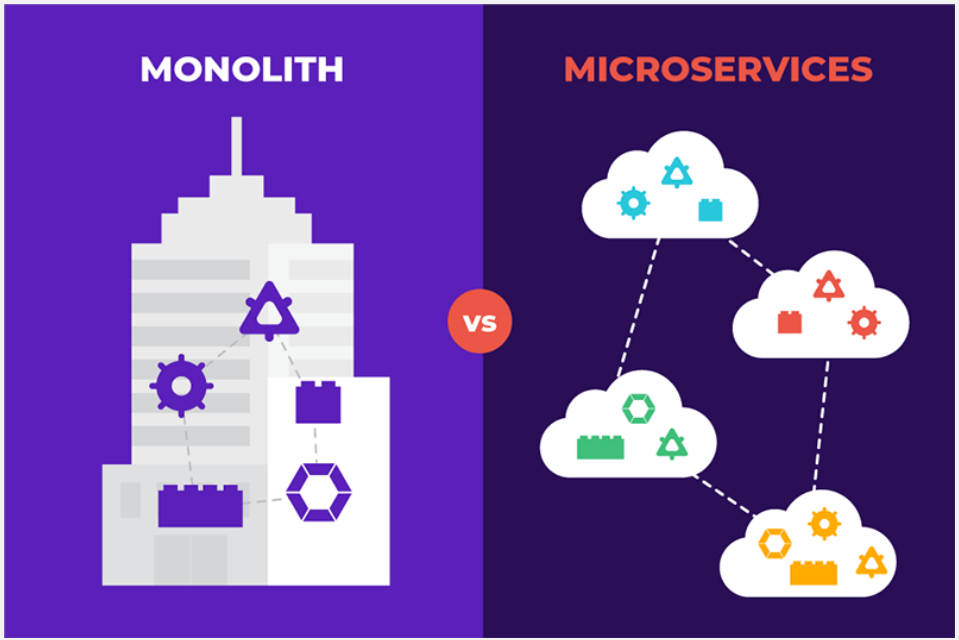
- Ciò che vedremo si applica tranquillamente a...
  - Applicazioni e/o servizi Windows che girano in background
  - Applicazioni Web e/o moduli Apache che ospitano Web API
  - «Monoliti» che racchiudono qualsiasi tipo di business logic

We ♡ Monoliths



Nessun monolite è stato maltrattato durante la realizzazione di queste slide.

Più monoliti



Prendiamo ciò che di buono viene dalle architetture basate su «microservizi».

- Usiamo i monoliti, ma rendiamoli isolati e indipendenti
- Diamo a ogni monolite una sola responsabilità
- Sfruttiamo i sistemi di messaggistica per le comunicazioni (es. NATS, RabbitMQ)
- Utilizziamo più linguaggi e tecnologie (se il know-how del team di sviluppo lo consente o lo incentiva)
- Semplifichiamo il deploy degli applicativi
- Aumentiamo la scalabilità e la potenza del nostro sistema software nel suo complesso

Tenere sotto controllo...

«La potenza è nulla senza controllo»

(Rocco Siffredi)

...in che modo?

- Generazione di log
  - Strumento essenziale di tracciamento
  - Facili da implementare

    LoggerPro 👉 https://github.com/danieleteti/loggerpro
  - Riportano prevalentemente eventi rilevanti per il business (di alto livello)
  - Se usati per monitoraggio, possono diventare troppo numerosi o rendere difficili le indagini successive
  - Non adatti a interventi preventivi
- Altri tool generici (es. Task Manager, Host Monitor, query WMI, ecc.)
  - Controllo troppo blando e generico
  - Strumenti non integrati

Ci vorrebbe...

- Monitoraggio in tempo reale
  - Lettura continua e assidua di variabili di interesse
  - Focus su elementi di basso livello (es. disponibilità di risorse macchina)
  - Verifiche estese a tutti i servizi (non solo business, ma anche architetturali)
  - Acquisizione di diverse tipologie di grandezze per analisi incrociate
  - Possibilità di archiviare dati recenti in modo rapido e massivo (e consultabile)
  - Eventuale indipendenza hardware del tool di monitoring

«Prevenire è meglio che curare!»

DelphiDay italian conference

```
begin
```

Prometheus

Che cos'è «Prometheus»?

- ➔ Tool di monitoraggio e alerting di sistemi software
  - ➔ Originariamente nato 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, crossplatform

- ❖ Sito ufficiale (con risorse e documentazione)
  👉 https://prometheus.io/

Cosa fa «Prometheus»?

- Raccoglie e archivia dati con un riferimento temporale (TSDB)
  - Contatta i servizi (es. tramite endpoint) verificandone lo stato
  - I dati vengono salvati unitamente a un identificativo e a un «timestamp»
  - E' possibile salvare 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

➔ Queste sono le caratteristiche principali di Prometheus:

- ➔ 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

(*) è disponibile anche una modalità «push» sfruttando i cosiddetti «gateway intermedi».

«Pull» dei dati

- La modalità «pull» nel recupero dei dati offre 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 usabile anche per processi che eseguono per un tempo limitato

Demo

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

Client Libraries

➔ Le «client libraries» sono librerie scritte in diversi linguaggi di programmazione

- ➔ 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 i linguaggi **Go, Java, Python, Ruby** (ufficiali)

C/C++, Dart, Elixir, Haskell, LUA, .NET/C#, Node.JS, Perl, PHP, R (non ufficiali)

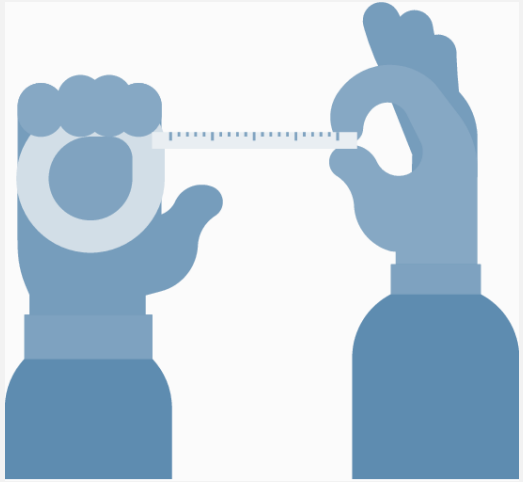
...e 🏛 Delphi? 😮

Le metriche

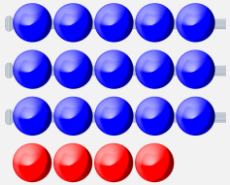
- Cosa sono le metriche?
  - Termine usato per identificare le misurazioni rilevate, associate a un riferimento temporale, e acquisite in modo continuativo nel tempo.

- 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

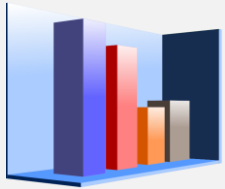
- ➔ Rappresenta un contatore sempre crescente (il valore può solo aumentare)
- ➔ Una volta azzerato all'avvio, può essere incrementato di una o più unità

Esempi: numero di attività completate, numero di richieste ricevute, conteggio 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

(*) potrebbero non essere disponibili nel client!

Demo

- ➔ Creiamo una applicazione Web basilare
- ➔ Configuriamo un nuovo job su Prometheus
- ➔ Esponiamo alcune metriche di esempio
- ➔ Ispezioniamo lo stato del job e le metriche raccolte usando il linguaggio integrato PromQL

Exporters

- Grazie agli exporters, può raccogliere dati da altri sistemi
  - Sono disponibili servizi per esportare metriche verso Prometheus
- E' possibile acquisire informazioni sul «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»
  - ➔ 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 (!)

Demo

➔ Testiamo la raccolta di metriche da un exporter

- Configuriamo (velocemente) il tool «Windows Exporter»

➔ Testiamo le funzionalità base degli alert

- Definiamo una regola per l'invio di un alert
- Spediamo l'alert via e-mail tramite SMTP

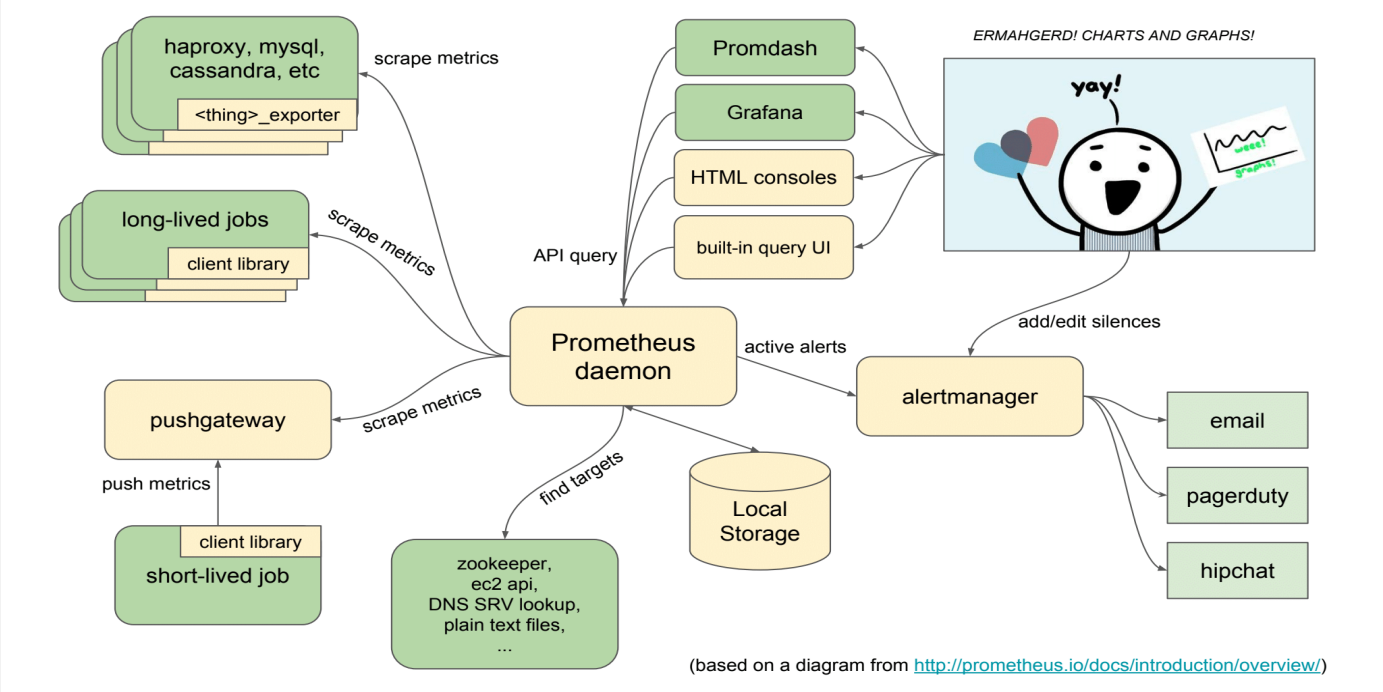
There’s more!

- 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

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

Scenario tipico



DelphiDay italian conference

Grafana

Che cos'è «Grafana»?

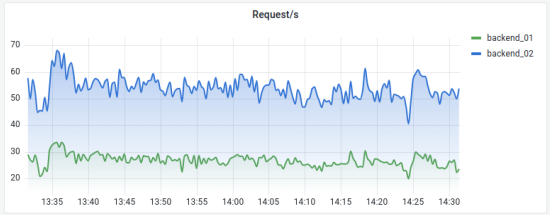
- Tool che consente di interrogare, visualizzare, generare avvisi e comprendere le tue metriche
  - Consente di recuperare metriche da diverse fonti dati
  - Supporta il recupero delle serie da Prometheus
  - Disegna dashboard progettate per il management
  - Sviluppato e gestito dalla community di Grafana Labs
  - Open source e rilasciato con licenza Apache 2.0

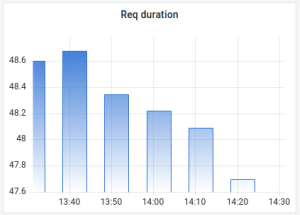
- Sito ufficiale (con risorse e documentazione)
  👉 https://grafana.com/oss/grafana/

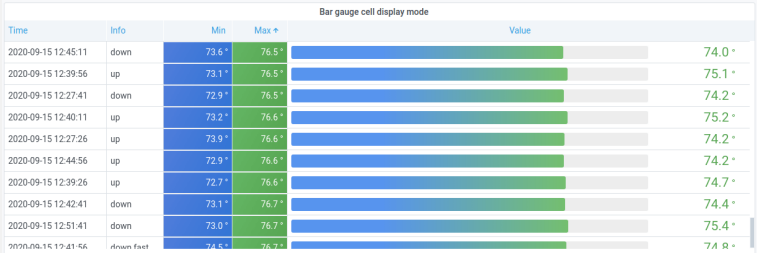
Gli elementi di Grafana

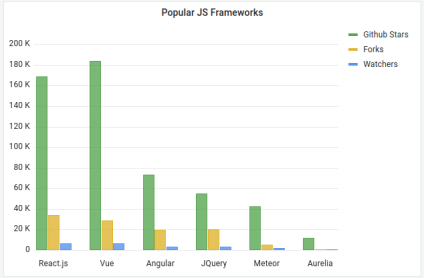
- ➔ Dashboard
  - ➔ Rappresenta una singola videata, composta da diverse rappresentazioni
  - ➔ Svolge la funzione di un cruscotto dedicato a un ambito specifico
- ➔ Panel
  - ➔ Riquadro (con titolo e descrizione) aggiunto alla dashboard
- ➔ Visualization
  - ➔ Grafici di varie tipologie che si possono inserire nel pannello
- ➔ Data Source
  - ➔ Origine dalla quale prelevare i dati da mostrare (Prometheus è una di queste)

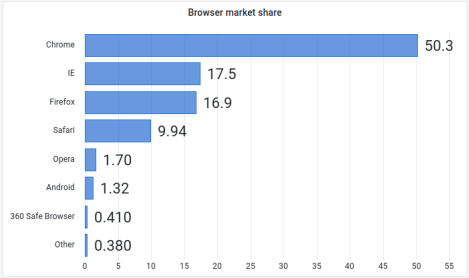
Visualization

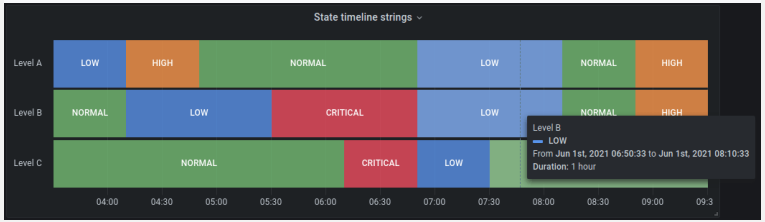


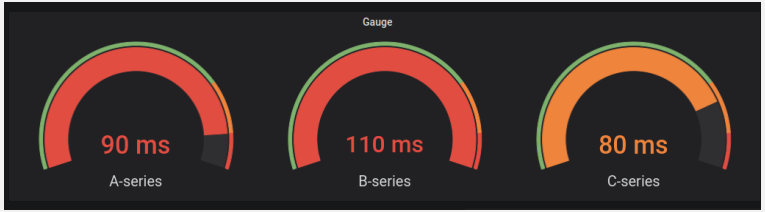


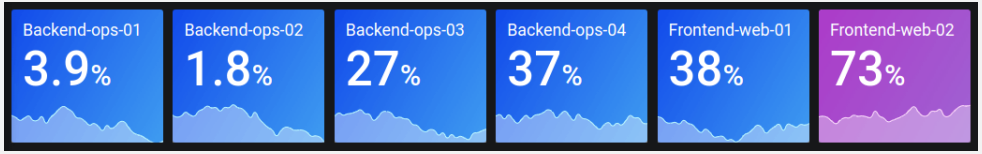












DelphiDay italian conference

Demo

- ➔ Creiamo una dashboard di esempio
- ➔ Aggiungiamo Prometheus come Data Source
- ➔ Progettiamo un nuovo pannello con un grafico di esempio
- ➔ Esploriamo le proprietà principali che possiamo personalizzare per ogni elemento a disposizione

Un demo vale più di mille parole... 

Altri spunti per l'uso

- ➔ Costruire dashboard rivolte al business
  - ➔ Estrarre dati dai database server dei propri software
  - ➔ Presentare visualizzazioni e proiezioni intelligibili
  - ➔ Monitorare i livelli di produttività e adesione agli SLA
- ➔ Dare seguito al «mantra» di Grafana
  - ➔ Promuovere una cultura basata sui dati

Conclusioni

Q & A

```
end;
```

DelphiDay
italian conference

ONLINE EDITION

THANK YOU

DelphiDay
italian conference