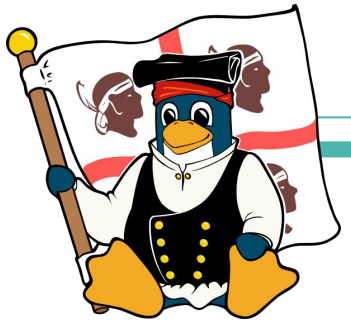
Monitoring con Prometheus

Matteo Dessalvi

LinuxDay 28 Ottobre 2017



LINUX DAY ITALIA

About me

- Sysadmin da circa 13 anni
- ~8 anni presso il Dip. di Fisica (UniCA)
- Attualmente sysadm per il gruppo HPC del GSI Helmholtz Centre for Heavy Ion Research (www.gsi.de)
- Proud member of GULCh since 2000

Sommario

- Perche’ monitoring?
- Sistemi di monitoring tradizionali
- Prometheus: caratteristiche, architettura, queries
- Vantaggi e svantaggi
- Riferimenti

Perche' il monitoring e' una attivita' essenziale

- Vogliamo sapere quando le cose vanno male (eventualmente contattare una persona per rimediare od evitare che la situazione peggiori).
- Essere in grado di effettuare il debug di un problema e possibilmente capire cosa lo ha generato e perche'.
- Analisi di un trend: ad esempio, quanti eventi dello stesso tipo si sono verificati in un determinato intervallo di tempo.
- Eventualmente utilizzare le informazioni del sistema di monitoring per altri sistemi (security, configuration management, ecc.).

Sistemi di monitoring tradizionali

Sistemi di monitoraggio tradizionali utilizzati solitamente su piattaforma Linux sono *Ganglia, Zabbix, Zenoss, Munin* (fra i piu' noti). *Nagios/Icinga* sono usati solitamente per alerting.

- Non c'e' nulla di sbagliato nei sistemi precedentemente elencati ma essi si focalizzano principalmente su un singolo host, non su servizi.
- Cosa succede se voglio aggiustare il monitoraggio su una grana piu' fine? Ad esempio da un singolo servizio (MySQL, Apache, ecc.) estrarre solo uno specifico gruppo (o sottogruppo) di metriche?

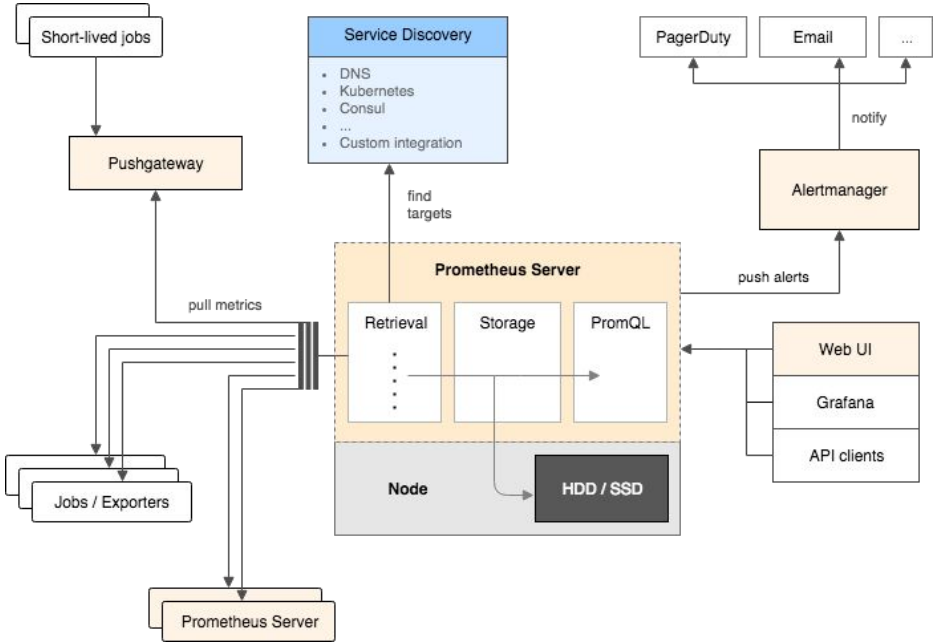
Prometheus short history

- Ispirato al sistema di monitoraggio interno usato da Google (‘Borgman’).
- Sviluppo partito da un gruppo di ex-Googlers nel 2012 presso Soundcloud.
- Lanciato pubblicamente nel 2015 come prodotto open source (Apache License version 2.0).
- Sviluppato in *GOlang* come linguaggio principale ma esistono clients che consentono l’accesso alle API in *Java*, *Python* e *Ruby*.

Prometheus: caratteristiche

- La raccolta dei dati avviene via **pull model** over HTTP.
- Un modello di gestione dei dati **multidimensionale** basato su serie temporali (*metric name and key/value pairs*).
- ***PromQL***: un linguaggio orientato principalmente alle queries su timeseries data.
- Prometheus non si basa su storage distribuito: ogni singola istanza e' autonoma (il server e' contenuto in un singolo binario) e salva i dati direttamente su storage locale.
- Supporto per dashboards basato su **Grafana**.

Prometheus Architecture



Prometheus: instrument your code

Prometheus ha sostanzialmente due approcci al monitoraggio:

1. Introdurre nel codice della propria applicazione delle funzionalita' atte ad esporre direttamente le metriche necessarie al monitoraggio (via HTTP).
2. Per servizi e/o applicazioni nelle quali non e' possibile intervenire direttamente Prometheus si basa sui cosiddetti ***Exporters*** ovvero singole applicazioni (standalone) che estraggono le metriche necessarie e le esportano attraverso entry points accessibili via HTTP.

Prometheus pull model

Prometheus controlla ad intervalli periodici tutti quei servizi che espongono metriche attraverso un entry point HTTP.

Differentemente da un sistema come *Ganglia*, dove sono i singoli demoni (*gmond*) a 'spingere' (**push**) le metriche verso uno o piu' server, nel caso di Prometheus e' il server che interroga ad intervalli periodici tutti i servizi che necessitano di essere monitorati e recupera (**pull**) i dati salvandoli nello storage locale.

Prometheus supporta comunque anche il modello push, se necessario.

Struttura di una metrica

Ogni serie di dati temporali viene **univocamente** identificata nel seguente modo:

<metric name> {<label name>=<label value>, ...}

- *<metric name>* specifica (in generale) una delle caratteristiche del sistema posto sotto osservazione.
- *<label name>=<label value>* sono coppie chiave-valore che consentono di gestire dati multidimensionali. Ogni combinazione delle labels per una stessa metrica identifica un particolare aspetto di essa (esempio: tutte le richieste HTTP con il metodo *POST* dirette verso */api/tracks* come handler).

PromQL: estrarre semplici componenti temporali

1) http_requests_total: restituisce tutte le serie temporali che riportano questa metrica.
2) http_requests_total{code="200",handler="/messages",method="get"} Restituisce tutte le serie temporali con la metrica '*http_requests_total*' e le labels '*code*', '*handler*' e '*method*'.
3) http_requests_total{job="apiserver", handler="/api/comments"}[5m] Una richiesta simile alla precedente ma ristretta in un intervallo di 5 minuti.

PromQL: funzioni ed operatori

1) rate(http_requests_total[5m]) : restituisce il tasso di variazione della metrica selezionata negli ultimi cinque minuti.
2) 1 - rate(node_cpu{cpu="cpu0",mode='idle',job='myjob'}[5m]) : quale percentuale di CPU time e' utilizzata per core.
3) topk(3, sum(rate(instance_cpu_time_ns[5m])) by (app, proc)) : estrae i tre maggiori consumatori di CPU time divisi per applicazione e tipo di processo.

Reference: PromQL query examples

Prometheus / Grafana

It’s demo time!

Cosa vedremo:

- Struttura del file di configurazione del server Prometheus.
- PromQL e Grafana dashboards.
- Query verso un exporter via HTTP.

Cosa puo' fare una istanza di Prometheus?

Monitoraggio per un cluster di calcolo HPC (GSI Kronos cluster):

- 552 bare metal nodes
- 13856 CPU cores
- Infiniband FDR Mellanox network

Una singola istanza di Prometheus (Virtual Machine) cosi' configurata:

- 6 CPU cores / 12 Gb RAM
- Storage VM: array di dischi SSD connessi direttamente al locale Hypervisor

Prometheus: cosa non puo' fare

- Supporto per gestire logs: Prometheus non fa aggregazione od offre strumenti di gestione dei logs.
- Prometheus non fornisce alcun tipo di storage persistente per il salvataggio dei dati (e' una scelta fatta in fase di progettazione).
- Prometheus non offre alcun tipo di dashboard nativamente: Grafana e' l'unica soluzione disponibile al momento.
- Prometheus non ha alcun supporto per *anomaly detection*. L'alert manager e' cio' che si avvicina di piu' ma non e' un sostituto per altri strumenti.

References

- Prometheus: sito web ufficiale
- Prometheus: Documentazione
- Prometheus Github repository
- Prometheus vs. {Graphite, InfluxDB, OpenTSDB, Nagios, Sensu}
- Awesome Prometheus (on Github)
- Brian Brazil blog (main developer)
- Grafana: web analytics interface

Thank you!

Domande?