Simple Docker Monitor

Progetto Gestione di Reti 2017/2018

Autore: **Sturba Gionatha**
Matricola: **531274**

Descrizione

Una delle recenti tecnologie sviluppatasi nell’ambito IT sono i Docker Container.

Un container è un’applicazione isolata (con tutte le sue dipendenze) eseguita in namespace separati (network namespace, file system, users, IPC).
I container condividono il kernel della macchina host (ovvero la macchina dove vi è allocato) e non hanno uno strato di virtualizzazione tramite hypervisor.

Docker , invece, è uno strumento che ha, come scopo principale, quello di rendere più facile la creazione, il deploy e l'esecuzione di applicazioni utilizzando i container.

Essendo, quindi, delle applicazioni vere e proprio che girano all'interno del sistema operativo, nasce la necessita' di dover monitorare questi container, sopratutto sull'impatto che hanno sulla macchino in cui girano (consumo Cpu,Ram,Disk,Network ecc…).

Il progetto consiste nello sviluppo di un sistema in grado di collezionare, processare ed aggregare una serie di metriche riguardanti i Docker container in esecuzione sulla macchina.

Esecuzione

NOTA: Il sistema funziona attualmente solo su *O.S* ***Unix-based***

Di seguito una descrizione su come eseguire una **demo in locale** del sistema.

(1) Attivare la Docker Rest API

1. Aprire il file ***/lib/systemd/system/docker.service***
2. Modificare la riga che inizia con *ExecStart* in questo modo: ***"ExecStart=/usr/bin/docker daemon -H fd:// -H tcp://0.0.0.0:4243"*** dove l'aggiunta fatta e' " *-H tcp://0.0.0.0:4243* ".
3. Salvare il file
4. Riavviare il Docker daemon con il comando " ***systemctl daemon-reload*** "
5. Riavviare il Docker Service con il comando " ***sudo service docker restart***"
6. Assicurarsi che la Docker Api sia attiva, con il comando: "***curl http://localhost:4243/version***". Dovreste avere un output del genere: "***{"Version":"1.11.0","ApiVersion":"1.23","GitCommit":"4dc5990","GoVersion":"go1.5.4","Os":"linux","Arch":"amd64","KernelVersion":"4.4.0-22-generic","BuildTime":"2016-04-13T18:38:59.968579007+00:00"}*** "

(2) Modificare il file */src/main/resources/collector.config*,

ed inserire sotto la voce ***DOCKER_REST_API_PORT,*** la porta su cui e' attiva la Docker API, di default e' la 4243 se si e' seguiti la procedura sopra descritta.

Gli altri campi, possono essere lasciati a quelli di default, a patto che si utilizzi lo script ***init_all.sh*** per eseguire il sistema,e non si voglia un altro tipo di configurazione personale.

(3) Eseguire lo script ***init_all.sh.***
E' necessario eseguire lo script trovandosi nella cartella dove lo script risiede e con i permessi di root(sudo).

Nota sullo script:
Lo script esegue i seguenti passi:

1. Crea, se non esistono, dei volumi di storage da usare per i container di Graphite e Grafana.
2. Fa partire Grafana, sottoforma di docker container (se l'immagine non esiste viene scaricata al momento), che utilizzera' la porta 3000 (localhost) per il proprio funzionamento.
3. Fa partire Graphite, sottoforma di docker container (se l'immagine non esiste viene scaricata al momento), che utilizzera' la porta 100 e la 2003 (localhost) per il proprio funzionamento.
4. Fa partire il collector, usufruendo della JVM.
5. Se il sistema e' partito correttamente, stampa a video dei comandi da eseguire se si vuole terminare il sistema.

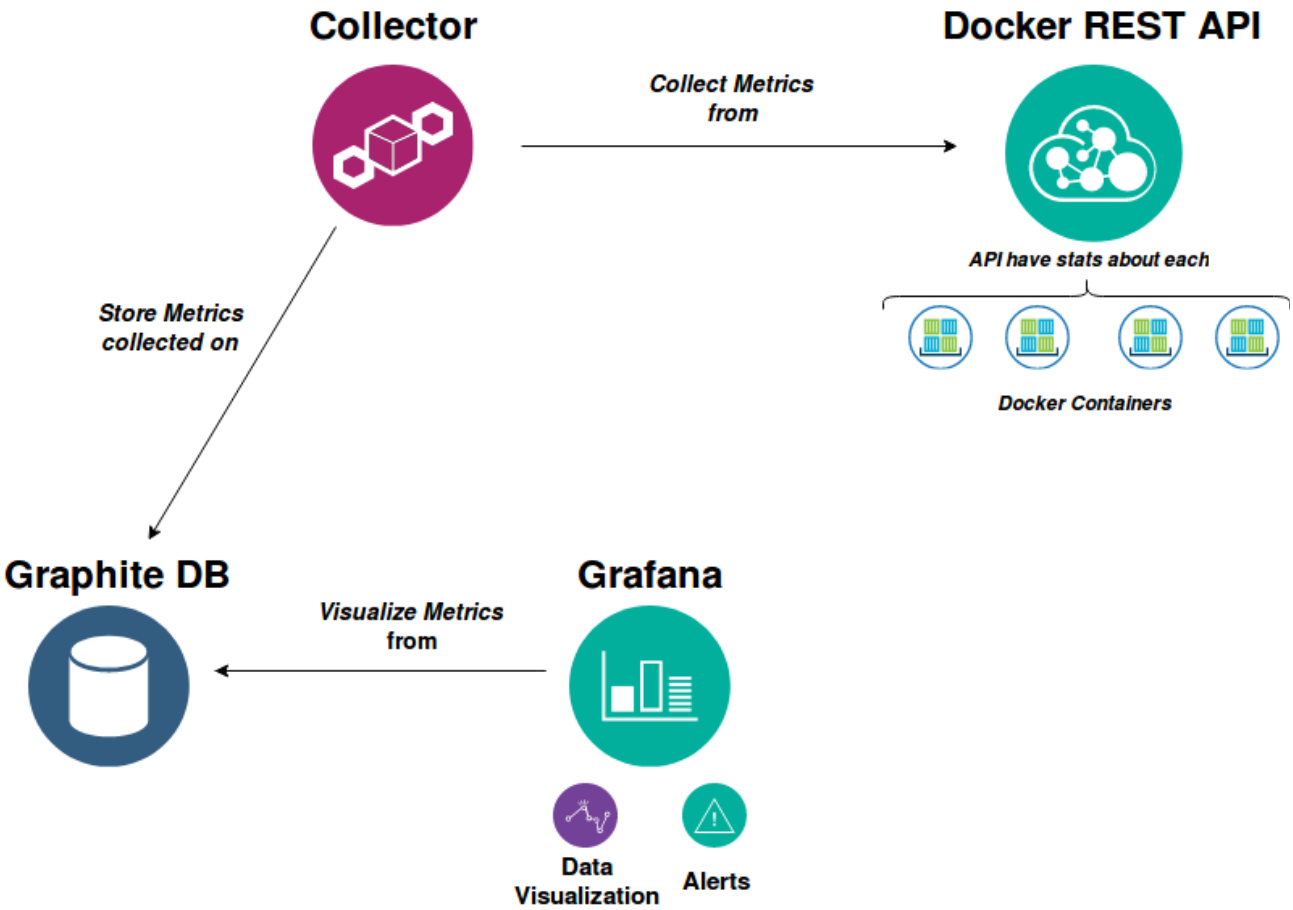
(4) Accedere a Grafana(localhost:3000) e seguire i seguenti passi:

- Nome utente: *admin* Password: *admin*
- Aggiungere un nuovo data source, dandogli come URL: *http://localhost:100*, Access: Server(Default), e poi salvare.

- Quando si creano i propri grafici o altro, tutte le metriche dei container si trovano sotto la voce: **Containers** (N*ota: dovreste trovare delle metriche riguardanti i 2 docker container in esecuzione Graphite e Grafana).*

Nota: Nella cartella *containers*, si trova uno script per far partire un o stress-container di prova.

Architettura del sistema



E’ stato sviluppato un collezionatore,in Java, che ha il compito di ricavare le metriche sui container in esecuzione, tramite la Docker Remote REST API.

Quest’ API, di tipo REST, dopo essere stata attivata sulla macchina in cui girera’ il sistema (guida attivazione API), conterra’ una serie di informazioni riguardanti ogni container in esecuzione sulla macchina.

Queste informazioni o statistiche, vengono collezionate dal Collector, tramite il protocollo REST, per poi essere processate in delle metriche piu’ affine (es: consumo in percentuale cpu, network speed on default interface, ecc..).

Una volta processate le metriche, quest’ultime vengono inviate e salvate all’interno di un Time Series Database, in questo caso Graphite.

Vi e’ poi un altro servizio, Grafana, in grado di visualizzare le metriche raccolte all’interno di Graphite, con opportuni grafici e con la possibilita’ di settare anche degli allerts.

Il Collector

Il Collezionatore di metriche sui container, e’ stato sviluppato per il progetto, mentre Graphite e Grafana vengono utilizzati come servizi di supporto al collezionatore.

E’ stato scritto in Java, ed ha un architettura multithread.

In questo caso abbiamo un thread (*collector.java*) che ha il compito di interrogare l’API per vedere quali container sono attivi in un determinato istante, per poterne poi ricavare delle metriche; operazione che si ripete ad

intervalli ogni tot secondi (*settando un timeout all'interno del file di configurazione).*
L'operazione di recupero informazioni e invio metriche al DB, riguardanti un particolare container,viene svolta pero' da un altro thread (M*etricContainerCollector.java*) all'interno di un pool, che e' gestito sempre dal collector(*collector.java*).

Una volta recuperate le informazioni su un particolare container, vengono processate delle metriche(*consumo CPU*, *consumo RAM*, *operazioni IO*, *network speed (tx,rx)*), che verrano poi inviate al DB gestito da Graphite,tramite una connessione di rete.

Graphite e Grafana

Come gia' accenato, sono stati usati 2 servizi di supporto al collector, Graphite e Grafana.

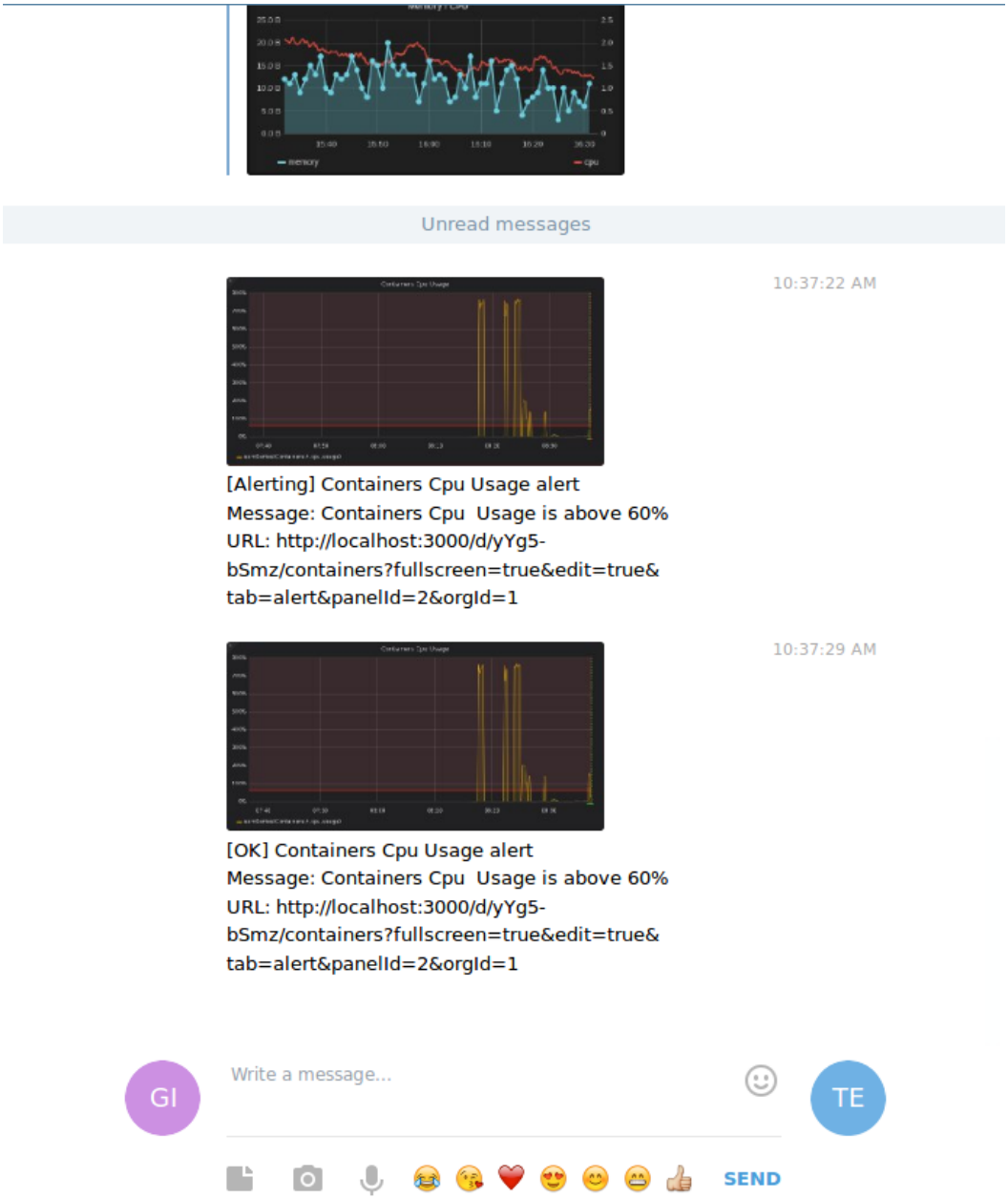
Ovviamente sono propedeutici al funzionamento del collector, quindi devono essere installati sulla macchina host (**nel caso del progetto, vengono fatti partire come 2 docker container separati, all'interno dello script *init_all.sh***).

Graphite viene utilizzato come DB per collezionare le metriche inviategli dal collector.
In genere utilizza la porta 2003 per ricevere le metriche, grazie al Carbon Metric daemon.

Grafana viene utilizzato per Data Visualization delle metriche, ed anche per settare degli allert al superamento di valori di sogli limite.
In genere e' raggiungibile sulla porta 3000.

Esempi di DashBoard di Grafana:

Esempio di Allert:



Una volta che il sistema funziona correttamente, e’ possibile accedere alla Web Interface di Grafana (su porta 3000) per visualizzare e creare i propri grafici, e settare gli allert per particolari situazioni critiche.