

RELAZIONE HOMEWORK#3-finale
DISTRIBUTED SYSTEMS AND BIG DATA

Studenti:

Nunzio Fornitto: 1000002901

Angelo Frasca : 1000067615

Professori:

Antonella Di Stefano

Giovanni Morana

Tutor:

Massimo Gollo

Abstract

In questo HW3, sono stati aggiunti: un server prometheus, che preleva i dati da due exporter. Un exporter è un thread in esecuzione nel server_grpc, gira sulla porta 8000, l'altro exporter è un thread in esecuzione nel datacollector, gira sulla porta 8002. Il server prometheus, fa periodicamente il pull dagli exporter e preleva le seguenti metriche:

Metriche prelevate dal server:

- 1) `grpc_request_count` (conta il numero di volte in cui viene chiamata la funzione `registerUser` del server) è una metrica di tipo `Caunter`
- 2) `grpc_response_time` (calcola il tempo di risposta della funzione `registerUser`) è una metrica di tipo `Gauge`

Metriche prelevate dal datacollector:

- 1) `save_stock_data_calls_total` (conta il numero di volte che la funzione, `savestockdata` è chiamata) è una metrica di tipo `Caunter`
- 2) `users_in_db` (conta il numero di utenti) è una metrica di tipo `Gauge`

Kubernetes (descrizione degli oggetti usati)

Di tutti i microservizi è stato fatto il deploy su Kubernetes. E' stato creato un cluster con kind, impostando un nodo control-plane e un nodo worker. Per accedere ai servizi dall'esterno del cluster è stato creato un “**Service**” (di tipo `nodeport`) che espone il `clientgrpc` all'esterno del cluster sulla porta **30000**. Ed è stata usata la funzionalità di **kind** per creare un port forwarding (`extraportmapping`). Questo permette di accedere alle funzionalità offerte dal `clientgrpc` tramite **localhost:5000**. Si è seguito lo stesso approccio per accedere ai dati che preleva dagli exporter prometheus server, quindi creando un altro `extraportmapping`, in particolare accedendo a

localhost:5001. Di ogni microservizio è stato creato il file di deployment, e ogni microservizio è stato inserito su un pod separato, i pod possono cumincare fra loro poiché per ognuno di essi è stato definito un “**Service**” offrendo un endpoint unico e fisso in caso di riavvii. L’unico pod contenente due container è quello relativo a Kafka e Zookeeper. Per quanto riguarda prometheus è stato usato il configMap per montare il file di configurazione di prometheus. Per mysql sono stati montati due volumi uno chiamato **mysql data** ed è un persistentVolumeClaim , quindi sopravvive alla morte del pod, e uno chiamato **init-script** ed è un configMap che contiene lo script di inizializzazione del database, quindi senza persistenza. Per il resto dei microservizi, quindi alert-system, alert-system-notifier, servergrpc e clientgrpc sono stati creati manifest di deployment ognuno con un service e un pod.

Diagramma architetturale dei nuovi componenti

