Documento di Build & Deploy per il Progetto

I. Requisiti

Per poter eseguire e deployare il progetto, è necessario aver installato i seguenti strumenti:

- **Docker**: necessario per creare e gestire i container dei microservizi.
- **Docker Compose**: necessario per orchestrare i container.
- **Kubernetes (Kind)**: per creare un cluster Kubernetes locale.
- **kubectl**: per interagire con il cluster Kubernetes.
- **Prometheus**: per raccogliere le metriche esportate dai microservizi.
- **Prometheus Exporters**: per esporre le metriche dai microservizi.
- **Python**: per implementare il codice che espone le metriche tramite Prometheus.

Inoltre, è necessario installare i seguenti:

- **prometheus_client**: necessario per esportare le metriche a Prometheus.
- **requests**: per inviare richieste HTTP tra i microservizi.
- **grpcio**: necessario per implementare la comunicazione tra client e server usando **gRPC**.
- **grpcio-tools**: necessario per generare il codice Python dal file `.proto`.
- **yfinance**: permette di ottenere dati finanziari (prezzi azionari, volumi, ecc.) da **Yahoo Finance**.
- **mysql-connector-python**: per connettere il progetto al database **MySQL** e gestire le query.
- **e-mail-validator**: verifica che gli indirizzi **e-mail** forniti dagli utenti siano validi.
- **confluent-kafka**: fornisce un'interfaccia Python per **Kafka**, utile per la gestione di flussi di dati in tempo reale.

Questi pacchetti sono definiti nei file `requirements.txt` del progetto.

II. Istruzioni da eseguire

1. **Clonare il repository:**

   - `Git clone https://github.com/MassiFino/HW3-DSABD.git`

2. **Creazione del Cluster Kubernetes**
   Per avviare un cluster Kubernetes, esegui il seguente comando:

   - `kind create cluster --name workshop --config kind-config.yaml`

3. **Creazione delle Immagini Docker**
   Esegui i seguenti comandi nelle rispettive cartelle per costruire le immagini Docker dei microservizi:

   - `docker build -t data-collector:latest .`
   - `docker build -t alert-notifier:latest .`
   - `docker build -t server:latest .`
   - `docker build -t telegram-bot:latest .`
   - `docker build -t alert-system:latest .`

4. **Caricamento delle Immagini Docker nel Cluster Kubernetes**
   Una volta costruite le immagini Docker, caricale nel cluster Kubernetes con i seguenti comandi:

   - `kind load docker-image data-collector:latest --name workshop`
   - `kind load docker-image server:latest --name workshop`
   - `kind load docker-image telegram-bot:latest --name workshop`
   - `kind load docker-image alert-system:latest --name workshop`
   - `kind load docker-image alert-notifier:latest --name workshop`

5. **Avvio dei Servizi in Kubernetes**
   Applicare i manifesti YAML per avviare i microservizi nel cluster:

   - `kubectl apply -f database.yaml`
   - `kubectl apply -f datacollector.yaml`
   - `kubectl apply -f kafka.yaml`
   - `kubectl apply -f server.yaml`
   - `kubectl apply -f telegram-bot.yaml`
   - `kubectl apply -f alert-system.yaml`
   - `kubectl apply -f alert-notifier.yaml`
   - `kubectl apply -f prometheus.yaml`

6. **Verifica lo Stato dei Pod**
   Puoi verificare lo stato dei pod con il comando:

   - `kubectl get pods`

7. **Caricamento del Backup del Database**

Per inizializzare e caricare un backup del database SQL nel cluster Kubernetes, è necessario seguire questi passaggi:

a) **Caricare il Backup nel Pod MySQL**
   Utilizza il comando `kubectl cp` per copiare il file di backup all'interno del pod MySQL:

   - `kubectl cp ./backup_db.sql mysql-0:/tmp/backup_db.sql`

b) **Accedere al Pod MySQL e Ripristina il Backup**
   Accedi al pod MySQL ed esegui il comando per ripristinare il backup:

   - `kubectl exec -it mysql-0 -- bash`

   - `mysql --binary-mode=1 -u root -p1234 yfinance_db < tmp/backup_db.sql`

V. Test e Debug

– Login

È possibile effettuare l'accesso utilizzando questo utente:

- masssifino@gmail.com

Saranno associati alcuni ticker all’utente e sarà possibile effettuare le seguenti operazioni:

1. **Aggiungi ticker**: Permette all'utente di aggiungere un nuovo ticker e decidere se inserire un valore di massimo e minimo per il monitoraggio dei ticker.
2. **Visualizza tutti i ticker**: Mostra tutti i ticker associati all'utente.
3. **Aggiorna un ticker**: Permette di sostituire un ticker esistente con uno nuovo decidere se inserire un valore di massimo e minimo per il monitoraggio dei ticker.
4. **Elimina un ticker**: Permette di rimuovere un ticker dalla lista dei ticker associati all'utente.
5. **Ottieni ultimo valore**: Permette all'utente di ottenere il valore più recente di un titolo finanziario associato a un ticker specificato.
6. **Ottieni media valori**: Calcola e mostra la media dei valori di un ticker specificato.
7. **Elimina utente**: Permette all'utente di eliminare il proprio account, rimuovendo tutte le informazioni associate, inclusi i ticker e i dati salvati.
8. **UpdateMinMaxValue**: Permette di aggiornare i valori massimo e minimo di un ticker esistente.
9. **Esci**: permette l'uscita del programma.

Per utilizzare il servizio e-mail bisogna inserire una e-mail esistente, altrimenti non si riceverà nessuna notifica.

- Per usufruire anche del bot Telegram, seguire i seguenti passaggi:
  1. Cercare su Telegram *alert_notifier1_bot* e premere start.
  2. Dopo averlo avviato scrivere getUpdates che permette al bot di ottenere l'id della chat.
  3. Successivamente, si aprirà una finestra di dialogo con il bot che vi chiederà di inserire la vostra e-mail per due volte, questo servirà per associare il *chat_id* all' e-mail nel database così da ricevere le notifiche in questa chat.
  4. Dopo aver eseguito questi passaggi si riceveranno periodicamente informazioni sui propri tickers.

L'e-mail deve corrispondere a quella inserita nel Database altrimenti non verrà associata. Il servizio Telegram è reso facoltativo.

– Registrazione

Per effettuare la registrazione è necessario fornire un'e-mail scritta nel formato corretto ovvero:

a. Nome utente:
- Può contenere lettere (a-z, A-Z), numeri (0-9), punti (.), trattini (-), e underscore (_).
- Non può iniziare né finire con un punto.
- Non può contenere spazi.

b. Chiocciola (@):
- Un singolo simbolo di "@" che separa il nome utente dal dominio.

c. Dominio:
- Deve contenere almeno un punto (".").

- Deve avere una parte di dominio di livello superiore (TLD), come .com, .org, .net, ecc.
- Le singole sezioni del dominio (prima e dopo ogni punto) devono essere composte da lettere e numeri, senza trattini all'inizio o alla fine.

Successivamente chiederà di inserire un ticker che dovrà essere riconosciuto da yahoo finance; infatti, è presente un controllo che verifica il corretto inserimento confrontandole con file .csv contenente tutti i ticker, ne consigliamo di inserire alcuni:

- AAPL – Apple Inc.
- MSFT – Microsoft Corporation
- GOOGL – Alphabet Inc. (Google)
- AMZN – Amazon.com, Inc.
- NFLX – Netflix, Inc.
- FB – Meta Platforms, Inc.
- DIS – The Walt Disney Company
- NVDA – NVIDIA Corporation
- BA – Boeing Company

– Monitoraggio con Prometheus

Il sistema di monitoraggio tramite **Prometheus** raccoglie metriche dai microservizi utilizzando gli **exporters**. Ogni microservizio espone due tipi di metriche principali:

- **GAUGE**: per misurare valori variabili, come il tempo di risposta.
- **COUNTER**: per misurare contatori incrementali, come il numero di richieste.

Inoltre, ogni metrica è etichettata con informazioni sul servizio e sul nodo che ospita il microservizio, così da poter essere filtrata facilmente in Prometheus.

Visualizzazione le Metriche
Le metriche raccolte da **Prometheus** possono essere visualizzate tramite l'interfaccia web di Prometheus. Accedi alle metriche al seguente indirizzo:

`http://localhost:30090`

Per visualizzare le metriche esposte dopo essersi recati all'indirizzo, è necessario effettuare delle query in base alla metrica da visualizzare.
Vengono mostrati di seguito alcuni esempi:
Nella figura 1 mostra la metrica `telegram_messages_sent_total` di tipo counter relativa al numero di messaggi telegram inviati, filtrata per il servizio alert-notifier. Mentre nella Figura 2 viene mostrato la metrica `email_sent_total` che monitora il numero cumulativo di e-mail inviate dal sistema, filtrata in base al tipo di metrica (alert_email), al servizio responsabile (alert-notifier) e al nodo specifico (worker). Il risultato è una visualizzazione che riporta il valore totale del contatore, ovvero il numero di e-mail inviate con successo fino al momento della query. Nella figura 3 mostra data_collector_ticker_processing_duration_seconds che permette di monitorare e analizzare il tempo necessario per gestire ogni ticker, filtrato in base alla metrica selezionata e al servizio. Nella figura 4 visualizza una metrica ottenuta dal node exporter. La query `rate(node_cpu_seconds_total{mode="user"}[1m]` estrae informazioni sull'utilizzo della CPU in modalità **user** nel tempo.

Figura 1: telegram_messages_sent_total , descrive il numero totale di messaggi Telegram inviati con successo

Figura 2: email_sent_total, descrive il numero totale di e-mail inviate con successo

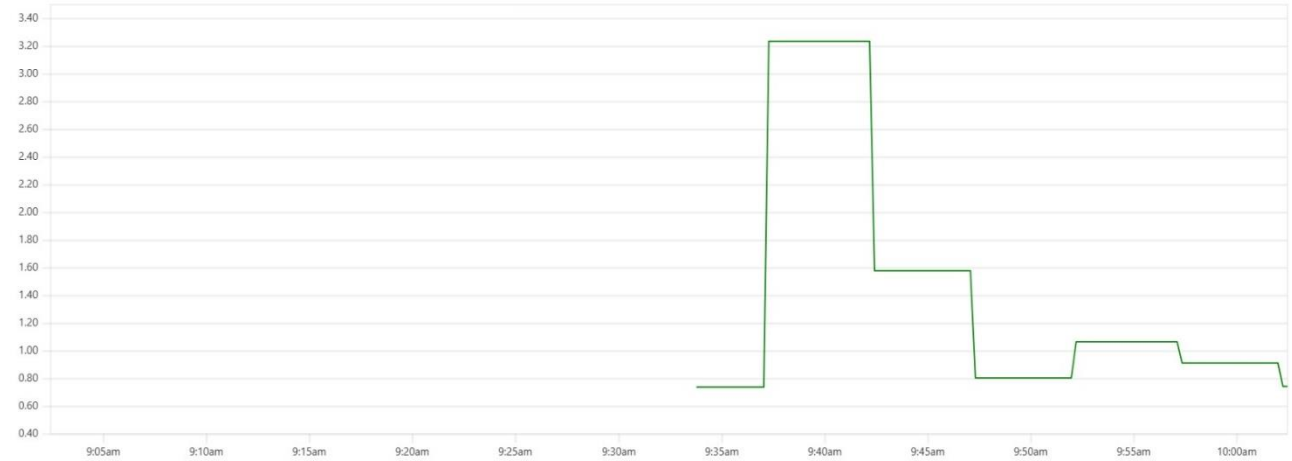

Figura 3: data_collector_ticker_processing_duration_seconds, decrive durata in secondi dell'elaborazione di un ticker nel sistema

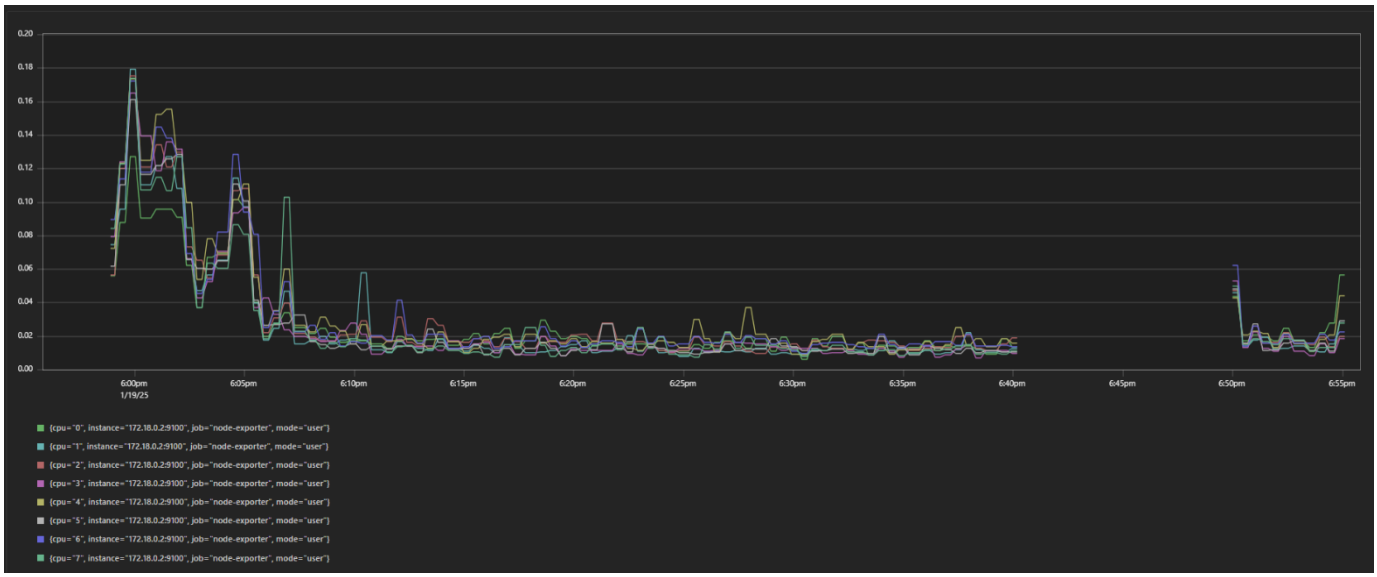

Figura 4: Metrica estratta dal Node Exporter che mostra l'utilizzo della CPU in modalità "user"

Documento redatto da:
Massimiliano Finocchiaro, Dario Rovito