GUIDA ALLA BUILD E DEPLOY

Homework 3

Sommario

1.	Guida alla Build e Deploy	. 2
1.1	Clonare la repository	. 2
1.2	Build e avvio dei container	. 2
1.3	Verificare che il sistema sia pronto	. 2
1.4	Avvio del client	. 2
1.5	Arrestare il sistema	.3
1.6	Comandi Utili per MySQL	.3
2.	Comandi Utili per Kafka	. 4
3.	Configurazione di Minikube	.5
3.1	Avvia Minikube	. 5
3.2	Crea un Namespace	. 5
3.3	Configurazione dell'Ambiente Docker	. 5
3.4	Build delle Immagini Docker	. 5
3.5	Caricamento dei Manifests Kubernetes	. 5
3.6	Verifica dello Stato dei Pods	. 5
3.7	Esporre il GRPC Server	. 5
4.	Monitoraggio	. 6
4.1	Metriche del Server	. 6
4.2	Metriche del Data Collector	.6
5.	Debugging	.6
5.1	Debug dei Pods	. 6

1. Guida alla Build e Deploy

Questo documento descrive i passaggi necessari per eseguire la build, il deploy e l'utilizzo del sistema distribuito.

1.1 Clonare la repository

Per iniziare, clonare la repository del progetto utilizzando il seguente comando:

git clone https://github.com/TanoBont/Homework2-DSBD_2425.git

Posizionarsi nella cartella del progetto con il comando:

cd Homework2-DSBD_2425

1.2 Build e avvio dei container

Prima di avviare i container, è necessario costruire le immagini Docker dei servizi. Utilizzare il comando:

docker-compose up --build

Questo comando esegue contemporaneamente:

- La build delle immagini.
- L'avvio dei container definiti nel file docker-compose.yml.

Nota importante:

Il file docker-compose.yml include un health check per verificare che il database sia completamente operativo prima di avviare gli altri container. Questo garantisce che il server gRPC e il data collector non tentino di connettersi al database prima che sia funzionante.

1.3 Verificare che il sistema sia pronto

Attendere qualche secondo dopo che i log si sono fermati per essere certi che il sistema sia completamente operativo.

1.4 Avvio del client

Per utilizzare il sistema, aprire un nuovo terminale e avviare il client con il seguente comando:

python client.py

Il client si connetterà al server gRPC avviato all'interno dei container. Sarà quindi possibile interagire con le funzionalità del sistema.

1.5 Arrestare il sistema

Per interrompere il sistema e rimuovere i container senza eliminare i dati persistenti (volumi), utilizzare i comandi:

Ctrl + c

(nel terminale dove è stato fatto il docker compose e, nel terminale dove è in run il client, per interrompere l'esecuzione)

e successivamente:

docker-compose down

Se si desidera rimuovere anche i volumi associati (e quindi eliminare i dati persistenti):

docker-compose down -v

1.6 Comandi Utili per MySQL

Entrare nel container del database:

docker exec -it db bash

Accedere a MySQL con il seguente comando:

mysql -u myuser -p

Inserire la password:

mypassword

Selezionare il database del sistema:

USE dsbd_db;

I nomi delle tabelle sono i seguenti:

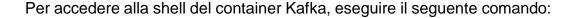
users;

stock data;

registration_messages

update_messages

2. Comandi Utili per Kafka



docker exec -it kafka bash

Per vedere la lista di tutti i topic Kafka presenti nel sistema, usare il comando:

kafka-topics --bootstrap-server kafka:9092 --list

Per **consumare i messaggi dal topic to-alert-system** e vedere i messaggi in tempo reale a partire dal primo messaggio, usare:

kafka-console-consumer --bootstrap-server kafka:9092 --topic to-alert-system --from-beginning

Per **consumare i messaggi dal topic to-notifier** e vedere i messaggi in tempo reale a partire dal primo messaggio, usare:

kafka-console-consumer --bootstrap-server kafka:9092 --topic to-notifier --from-beginning

Per **inviare messaggi al topic to-alert-system**, usare il comando kafka-consoleproducer. Questo comando permetterà di scrivere messaggi direttamente dalla shell che saranno inviati al topic specificato:

kafka-console-producer --bootstrap-server kafka:9092 --topic to-alert-system

Dopo aver eseguito il comando per il producer Kafka, inserire i messaggi da inviare in modalità interattiva.

> inserire messaggio da inviare

Ogni volta che viene premuto **Enter**, il messaggio viene inviato immediatamente al topic to-alert-system.

3. Configurazione di Minikube

3.1 Avvia Minikube

minikube start

3.2 Crea un Namespace

kubectl create namespace dsbd-namespace

3.3 Configurazione dell'Ambiente Docker

Per targhettare docker-env e costruire le immagini all'interno di Minikube:

minikube docker-env | Invoke-Expression

3.4 Build delle Immagini Docker

Esegui i seguenti comandi per costruire le immagini dei container personalizzati:

docker build -t grpc-server:latest -f server/Dockerfile.server .

docker build -t data-collector:latest -f data_collector/Dockerfile.datacollector .

docker build -t cp_alertsystem:latest -f alertSystem/Dockerfile.cp_alertsystem .

docker build -t c_alertnotifiersystem:latest -f alertSystem/Dockerfile.c_alertnotifiersystem .

3.5 Caricamento dei Manifests Kubernetes

Carica i manifest nella namespace:

kubectl apply -f ./manifests/ -n dsbd-namespace

3.6 Verifica dello Stato dei Pods

Controlla lo stato dei pods e attendi che siano tutti in stato Running prima di procedere:

kubectl get pods -n dsbd-namespace

3.7 Esporre il GRPC Server

Per esporre la porta del server e utilizzarlo con il client:

kubectl port-forward svc/grpc-server 18072:18072 -n dsbd-namespace

4. Monitoraggio

4.1 Metriche del Server

Esporre la porta del GRPC Server per monitorare le metriche:

kubectl port-forward svc/grpc-server 8000:8000 -n dsbd-namespace

 Accedi alle metriche dal browser: <u>http://localhost:8000/metrics</u>

4.2 Metriche del Data Collector

Esporre la porta del Data Collector per monitorare le metriche:

kubectl port-forward svc/data-collector 8000:8000 -n dsbd-namespace

 Accedi alle metriche dal browser: http://localhost:8000/metrics

5. Debugging

5.1 Debug dei Pods

Controlla lo stato dei pods:

kubectl get pods -n dsbd-namespace

Per vedere i log di un pod specifico:

kubectl logs "<nome-pod>" -n dsbd-namespace

Per descrivere un pod e identificare eventuali errori:

kubectl describe pod "<nome-pod>" -n dsbd-namespace