Virtual Network and Cloud Computing - Project Presentation

Del corso del

Prof. Gialuca Reali

Venturi Marco, 346951

Perugia, Anno Accademico 2023/2024
Università degli Studi di Perugia
Corso di laurea magistrale in Ingegneria Informatica e Robotica
Curriculum Data Science
Dipartimento di Ingegneria

A.D. 1308
unipg
DIPARTIMENTO
DI INGEGNERIA

Contents

1. Introduzione

Coffee Break è un progetto che mira a sviluppare un applicazione web predisposta al rilascio tramite piattaforma Docker o Kubernetes. Questo servizzio web è composto da un web server progettato in Flask con l'obbiettivo di classificare, sulla base delle informazioni inerenti alla bevanda che si vuole bere, che tipo di bevanda contenente caffeina si sta bevendo. Il web server ha anche il compito di immagazzinare i dati in un database Cassandra, con l'obbiettivo di poter analizzare in futuro i dati raccolti. Il modello che permette le predizioni è stato realizzato tramite il supporto del dataset Caffeine Content of Drinks fornito da Kaggle, il modello è poi stato salvato ed esportato all'interno del web server tramite la libreria python pickle.
Come già anticipato Coffee Break è stato sviluppato per entrambe le piattaforme, Docker e Kubernetes e è possibile consultarle entrambe in questa repository di GitHub.

2. Tecnologie di Virtualizzazione

2.1 Immagini

Per quanto riguarda le immagini utilizzate per stanziare i container all'interno di Docker e Kubernetes, si sono utilizzate prevalentemente delle immagini custom. Le immagini presenti sono:

- webservice-app: L'immagine contiene un immagine python 3.9 a cui sono stati aggiunti i file del modello, il file main appartenente a flask e una cartella template con i file HTML. La porta esposta è la 5000 .
- caffeine_cassandra: immagine custom di bitnami/cassandra utilizzata per impostare le variabili di ambiente. La porta esposta è 9042.
- cassandra_init: immagine custom di bitnami/cassandra utilizzata per precaricare i file cql utilizzati per la configurazione del cassandra db principale. è usato solo nel rilascio Docker .
- nginx: immagine utilizzata per eseguire la configurazione di caffeine-cassandra db all'interno di Kubernetes e per preliminari fasi di test.

2.2 Docker

Per la realizzazione del rilascio docker è stato utilizzato lo strumento dockercompose. Nel file yaml sono stati istanziati tre container, uno per il servizio web e due per cassandra. Nel primo container si è definito: il network a cui appartengono i container (caffeine_network), variabili di ambiente per la connessione al database, le porte, il comando di avvio del file main.py e la dipendenza dal container cassandra. Per quanto riguarda il container cassandra, ha differenza del precedente container, si ha solo il montaggio di un volume per permettere la persistenza dei dati al db. Il Terzo container invece, chiamato cassandra_init, lancia un comando di bash per permettere la configurazione di inizializzazione al container cassandra.

2.3 Kubernetes

Il rilascio di kubernetes è composto da quatro file yaml di cui tre principali. Il primo file riguarda la definizione dell'oggetto Kubernetes PersistentVolumeClaim, oggetto che si occupa della connessione del volume fisico al Pod/oggeto che lo richiede. In questo caso si è scelto di non generare un oggetto Persistent Volume poichè sembrerebbe che, se si sviluppa su windows 11, i persistent volume non sono persistenti.

Il secondo oggetto creato è lo StatefulSet. Lo StatefulSet è stato definito per gestire il Pod contenente il database cassandra, questo perchè cosi è possibile mantenere lo stato persistente del database e dare un individualità al pod. Mentre, il service di cassandra definito all'interno dello stesso file yaml di tipologia ClusterIp:None, che espone il pod sulla porta 9042.

Il terzo oggetto definisce il web server come oggetto di tipo Deployment che gestiste un ReplicaSet con fattore di replicazione due. Il service espone le repliche sulla porta 5000 e ha come tipo di service il LoadBalancer.

Tutti gli yaml con immagine custom sono stati settati con imagePullPolicy:Never in modo che Kubernetes cerchi le immagini nel repository locale di Docker Desktop e non sul Docker Hub. L'ultimo oggetto definisce un semplice pod nginx.

2.4 Riflessioni sulle Tecnologie usate

L'implementazione di entrambe le tecnologie mi ha permesso di capire le potenzialita e i limiti dell'utilizzo della sola containerizzazione. In più è stato molto utile per capire come creare e gestire le varie immagini in modo da poterle includere direttamente all'interno di Kubernetes.

Il progetto all'inizio aveva l'obbiettivo di implementare un tokenring di tre nodi cassandra come database. Nel momento della progettazione su Docker mi sono reso conto però che il cluster cassandra non avrebbe sfruttato al 100% la sua potenzialità, per la mancanza della propietà di self-healing. Con Kubernetes, grazie alla sua propietà di self-healing, che permette di riallocare i POD gia terminati e quindi sfruttare al massimo le potenzialità di un cluster Database. Questo in seguito non è stato implementato per motivi di modifica dell'obbiettivo.

3. Possibili Implementazioni

3.1 Kubernetes

Sarebbe un ottima pretica quella di utilizzare oggetti di tipo Secret in modo da poter certificare con certificati TLS le informazioni sensibili. In più sarebbe utile la definizione di un oggetto Job che in caso di crash del o dei databse possa rigenerare gli schemi dei database in modo autonomo.

3.2 Cassandra

Si potrebbe definire un cluster di nodi cassandra con l'implementazione di un cassandra token ring in modo da poter migliorare la replicazione dei dati.