Build & Deploy Documentation

Introduzione

Weatherfy è un applicativo pensato per lo sviluppo e il deployment distribuito. Nucleo cardine del progetto è quello di realizzare un'architettura che permetta di ottenere, tramite API REST, previsioni meteo giornaliere. I possibili utenti possono "iscriversi" a delle notifiche personalizzate in base a delle condizioni meteo scelte (constraints).

Questo documento ha come scopo quello di spiegare come affrontare il deployment del nostro applicativo.

1. Requisiti di Sistema

Per quanto riguarda i requisiti di sistema da affrontare, per poter effettuare la build e il deploy, risulta necessaria l'installazione del docker engine (si consiglia docker desktop essendo lo strumento usato da noi).

Requisito fondamentale per il corretto deploy dell'applicativo è avere almeno 20 GB di spazio per le immagini. Ciò è dovuto alla presenza di 20 immagini da containerizzare nel nostro applicativo di cui 12 create con il metodo build nel file .yml.

Di queste 20 immagini:

- 1 è di mysql
- 1 di mongoDb
- 1 per il running di zookeeper
- 3 per il running di 3 broker
- 1 per prometheus
- 1 per SLA manager
- 12 sono immagini che eseguono i nostri microservizi che definiscono l'applicativo.

2. Ambiente di Sviluppo

Come ambiente di sviluppo si è usato Windows 10 e di conseguenza riteniamo opportuno consigliarlo per la build e il deploy. Per l'esecuzione di Docker consigliamo WSL2 o HyperV, seppur si riscontri con quest'ultimo una più elevata latenza.

3. Build e Deploy dell'Applicazione

La build e il deploy dell'applicativo è stata svolta tramite DockerCompose e apposito file .yml, che gestisce il build delle immagini e le relazioni tra di esse (condizioni di restart e avvio dei vari container). Ogni container è creato a partire da un'immagine che esegue uno specifico microservizio.

Passi per la build e il deploy:

1. Tramite cmd andare nella directory weatherfy-deployment
2. Avviare docker desktop o in ogni caso l'engine
3. Eseguire nel prompt: docker-compose -f ./ docker_compose_weatherfy.yml up
4. A questo punto l'applicativo è pronto per essere usato

0. Se si volesse terminare l'esecuzione : -f ./ docker_compose_weatherfy.yml down

Per monitorare il deployment si deve andare su docker desktop e verificare che tutti i container ad eccezione di kafka_setupper, container dedito alla cancellazione dei topic Kafka presenti e l'inizializzazione dei topic necessari all'applicativo. Una volta che kafka_setupper ha terminato, si può dare inizio all'esecuzione continuativa degli altri container.

4. Risoluzione dei Problemi

In fase di deployment abbiamo riscontrato un grave problema, che ci ha impedito di testare l'applicativo su Docker come avremmo voluto.

Ci siamo trovati impossibilitati ad eseguire l'applicazione intera a causa delle scarse risorse hardware dei nostri computer, costringendoci al test di non più di 2 immagini per volta.

Ci riserviamo la facoltà del remoto dubbio, di aver impostato erroneamente dei parametri di configurazione di WSL2 e/o HyperV, portandoci ad avere un risultato fallimentare nel deployment.

5. Riferimenti

L'unico riferimento che consigliamo di consultare per il deployment è la relazione, dove sono spiegate tutte le fasi e operazioni dell'applicativo.

5. Kubernetes

Il problema della scarsità delle risorse hardware può essere risolto utilizzando Kubernetes (k8s). Questo permette di distribuire orizzontalmente i conteiner mediante l'utilizzo dei pod.

Nella repository del progetto è stato creato uno scheletro di deployment all'interno della cartella k8s. Tale cartella contiene un file manifest k8s, contenente tutti i deployment e service necessari per il progetto

In alternativa si sono create delle cartelle specifiche che contengono tutti i service, i deployment e uno statefulService.

Qualora si volesse proseguire per l'ipotetico deploy tramite k8s, può essere eseguito uno dei due comandi:

- kubectl apply -f k8s.yml
    a. Esegue il comando per tutti i file manifest di deployments, dei services e degli statefull sets
- Kubectl apply -f <nome del file manifest>
    a. Esecuzione manuale per ogni file manifest

Successivamente all'applicazione del manifest al proprio cluster, sarà possibile controllare lo stato e gli eventuali errori tramite:

1)kubectl get pods

2)kubectl describe pod <nome_pod_da_controllare>