Esercitazione Docker Compose

Ecco un esempio di file docker-compose.yml che definisce un'infrastruttura composta da due servizi: un server web Nginx e un'applicazione web Python basata su Flask, entrambi eseguiti all'interno di container Docker:

file.yaml

```yaml
version: '3.8'
services:
  web:
    build: ./web
    ports:
      - "8080:8080"
    depends_on:
      - app
  app:
    build: ./app
    environment:
      - FLASK_ENV=development
    ports:
      - "5000:5000"
```

In questo esempio, il file YAML definisce due servizi: web e app. Entrambi i servizi vengono costruiti da immagini Docker definite nelle directory ./web e ./app.

Il servizio web espone la porta 8080 del container all'esterno, consentendo di accedere al server web Nginx tramite il browser all'indirizzo http://localhost:8080. Inoltre, il servizio dipende dal servizio app, che deve essere avviato prima del servizio web.

Il servizio app espone la porta 5000 del container all'esterno, consentendo di accedere all'applicazione web Python basata su Flask tramite il browser all'indirizzo http://localhost:5000. Inoltre, il servizio definisce una variabile d'ambiente FLASK_ENV con valore development.

Per avviare i container in base alla configurazione definita nel file docker-compose.yml, è sufficiente eseguire il comando docker-compose up nella stessa directory del file. Docker

Compose si occuperà di creare i container necessari e di avviare i servizi in modo coordinato.

Le keyword principali utilizzate per scrivere un file docker-compose.yml sono:

· version: specifica la versione di Docker Compose utilizzata nel file
· services: definisce i servizi che costituiscono l'applicazione multi-container
· build: indica come costruire l'immagine Docker del servizio
· image: indica l'immagine Docker utilizzata dal servizio
· ports: definisce le porte esposte dal servizio e la loro mappatura alle porte dell'host
· volumes: specifica i volumi da montare nei container
· environment: definisce le variabili d'ambiente del servizio
· depends_on: indica le dipendenze tra i servizi, ovvero l'ordine in cui devono essere avviati
· networks: definisce le reti utilizzate dai servizi

Ci sono anche altre keyword disponibili, come deploy per specificare le opzioni di deployment, configs per le configurazioni, e così via. Tuttavia, le keyword elencate sopra sono le più comuni e sono sufficienti per definire la maggior parte delle applicazioni multi-container.

Elenco completo comandi:

- version: specifica la versione di Docker Compose utilizzata nel file
- services: definisce i servizi che costituiscono l'applicazione multi-container
- build: indica come costruire l'immagine Docker del servizio
- image: indica l'immagine Docker utilizzata dal servizio
- command: definisce il comando da eseguire all'avvio del servizio
- entrypoint: definisce l'entry point del container
- container_name: assegna un nome specifico al container
- environment: definisce le variabili d'ambiente del servizio
- env_file: specifica il file di configurazione delle variabili d'ambiente
- volumes: specifica i volumi da montare nei container
- ports: definisce le porte esposte dal servizio e la loro mappatura alle porte dell'host
- networks: definisce le reti utilizzate dai servizi
- depends_on: indica le dipendenze tra i servizi, ovvero l'ordine in cui devono essere avviati

- links: definisce le dipendenze tra i container, ovvero l'ordine in cui devono essere avviati
- restart: definisce il comportamento di restart del servizio in caso di errore
- deploy: specifica le opzioni di deployment per il servizio
- configs: definisce le configurazioni utilizzate dal servizio
- secrets: definisce i secret utilizzati dal servizio
- healthcheck: definisce il controllo di salute del servizio
- logging: definisce le opzioni di logging per il servizio
- extends: estende la configurazione di un servizio esistente

Tutorial

Here is a step-by-step guide on how to configure a Cassandra cluster with 3 nodes using Docker Compose:

Create a Docker Compose file: Create a new file called docker-compose.yaml and add the following content:

✓ **ANSWER**

```yaml
version: '3'

services:
  cassandra1:
    image: cassandra:latest
    container_name: cassandra1
    ports:
      - "9042:9042"
    environment:
      - CASSANDRA_SEEDS=cassandra1,cassandra2,cassandra3
      - CASSANDRA_CLUSTER_NAME=docker-cassandra
      - CASSANDRA_NUM_TOKENS=256
      - CASSANDRA_DC=dc1
      - CASSANDRA_RACK=rack1
    volumes:
      - cassandra1-data:/var/lib/cassandra/data
      - cassandra1-logs:/var/log/cassandra
```

```
  cassandra2:
    image: cassandra:latest
    container_name: cassandra2
    ports:
      - "9043:9042"
    environment:
      - CASSANDRA_SEEDS=cassandra1,cassandra2,cassandra3
      - CASSANDRA_CLUSTER_NAME=docker-cassandra
      - CASSANDRA_NUM_TOKENS=256
      - CASSANDRA_DC=dc1
      - CASSANDRA_RACK=rack1
    volumes:
      - cassandra2-data:/var/lib/cassandra/data
      - cassandra2-logs:/var/log/cassandra

  cassandra3:
    image: cassandra:latest
    container_name: cassandra3
    ports:
      - "9044:9042"
    environment:
      - CASSANDRA_SEEDS=cassandra1,cassandra2,cassandra3
      - CASSANDRA_CLUSTER_NAME=docker-cassandra
      - CASSANDRA_NUM_TOKENS=256
      - CASSANDRA_DC=dc1
      - CASSANDRA_RACK=rack1
    volumes:
      - cassandra3-data:/var/lib/cassandra/data
      - cassandra3-logs:/var/log/cassandra

volumes:
  cassandra1-data:
  cassandra2-data:
  cassandra3-data:
  cassandra1-logs:
  cassandra2-logs:
  cassandra3-logs:
```

This configuration file defines three Cassandra services, each with a unique container name and port mappings. Each service is configured with the same cluster name and

seed nodes to form a single cluster. The data directories and log directories are mapped to separate volumes to persist data across container restarts.

Start the cluster: From the directory where the docker-compose.yaml file is located, run the following command to start the Cassandra cluster:

ANSWER:

```
docker-compose up
```

This command will start the Cassandra containers in detached mode, meaning they will run in the background.

Verify the cluster status: Once the containers have started, you can verify the cluster status by connecting to one of the nodes and checking its status. For example, to connect to cassandra1, run the following command:

ANSWER:

```
docker exec -it cassandra1 cqlsh
```

This will start a CQL shell session inside the cassandra1 container. From here, you can check the status of the cluster by running the following command:

```
docker exec cassandra3 nodetool status
```

This will display the status of the cluster and all its nodes.

FROM HERE YOU CAN FOLLOW THE STEPS SEEN IN THE PREVIOUS TUTORIAL ABOUT CASSANDRA