Docker

Docker è una piattaforma open-source che facilita la creazione, la distribuzione e l'esecuzione di applicazioni in contenitori. I contenitori consentono di impacchettare un'applicazione insieme a tutte le sue dipendenze, garantendo che l'applicazione funzioni in modo coerente in qualsiasi ambiente. Di seguito sono riportati i concetti e le operazioni fondamentali relativi a Docker:

Concetti Fondamentali

1. **Contenitore:**
    - Un contenitore è un'unità standardizzata di software che racchiude l'applicazione e tutte le sue dipendenze, inclusi il codice, le librerie e le configurazioni.

2. **Immagine:**
    - Un'immagine Docker è uno snapshot leggero e eseguibile di un sistema, che include il codice dell'applicazione, le librerie e altre dipendenze necessarie per eseguire l'applicazione.

3. **Dockerfile:**
    - Un Dockerfile è un file di testo che contiene le istruzioni per la costruzione di un'immagine Docker. Specifica le dipendenze, le configurazioni e i comandi necessari per creare l'immagine.

4. **Registro Docker (Docker Hub):**
    - Il registro Docker è un servizio che ospita immagini Docker. Docker Hub è il registro pubblico di Docker, ma è possibile utilizzare registri privati per immagini personalizzate.

Operazioni Principali

1. **Installazione di Docker:**
    - Scarica e installa Docker sul tuo sistema operativo seguendo le istruzioni specifiche per il tuo ambiente: https://docs.docker.com/get-docker/

2. **Verifica dell'Installazione:**
    - Dopo l'installazione, verifica che Docker sia correttamente installato eseguendo il comando:

```
docker --version
```

3. **Creazione di un Dockerfile:**
    - Crea un file chiamato `Dockerfile` nel tuo progetto e definisci le istruzioni necessarie per creare l'immagine Docker.

4. **Costruzione dell'Immagine:**
    - Naviga nella directory del progetto contenente il Dockerfile e esegui il comando per costruire l'immagine:

```
docker build -t nome_immagine:tag .
```

5. **Esecuzione di un Contenitore:**
    - Dopo la costruzione dell'immagine, esegui un contenitore utilizzando il comando:

```
docker run -d -p porta_locale:porta_contenitore nome_immagine:tag
```

6. **Gestione dei Contenitori:**

   - Puoi visualizzare i contenitori in esecuzione con il comando:

     ```
     docker ps
     ```

     Puoi fermare un contenitore in esecuzione con:

     ```
     docker stop ID_contenitore
     ```

7. **Push e Pull da Docker Hub:**

   - Puoi condividere le tue immagini su Docker Hub eseguendo i comandi:

     ```
     docker login
     docker push nome_utente/nome_immagine:tag
     ```

     Per scaricare un'immagine da Docker Hub:

     ```
     docker pull nome_utente/nome_immagine:tag
     ```

8. **Pulizia del Sistema:**
    - Puoi rimuovere immagini e contenitori non utilizzati con i comandi:

```
docker system prune
```

Ecco un esempio di Dockerfile per un'applicazione PHP basata su un server web Apache:

```
# Usa un'immagine di base con PHP e Apache
FROM php:7.4-apache

# Imposta la directory di lavoro nell'immagine
WORKDIR /var/www/html

# Copia i file dell'applicazione nella directory di lavoro
COPY . .

# Esponi la porta 80 (porta predefinita di Apache)
EXPOSE 80

# Opzionale: Installa eventuali dipendenze PHP necessarie
# RUN apt-get update && apt-get install -y nome_pacchetto

# Comando di avvio del server Apache
CMD ["apache2-foreground"]
```

In questo esempio:

- `FROM php:7.4-apache` : Utilizza un'immagine di base con PHP e il server web Apache

```
# Comando di avvio dell'applicazione
CMD ["npm", "start"]
```

In questo esempio:

- `FROM node:14` : Utilizza un'immagine di base con Node.js.
- `WORKDIR /app` : Imposta la directory di lavoro nell'immagine.
- `COPY . .` : Copia i file dell'applicazione (presupponendo che il Dockerfile si trovi nella stessa directory dell'applicazione) nella directory di lavoro dell'immagine.
- `RUN npm install` : Installa le dipendenze dell'applicazione.
- `EXPOSE 3000` : Espone la porta 3000, che è la porta predefinita su cui Express di solito ascolta.
- `CMD ["npm", "start"]` : Specifica il comando di avvio dell'applicazione.

Puoi utilizzare un file `docker-compose.yml` simile all'esempio precedente per semplificare il processo di avvio del contenitore per l'applicazione JavaScript. Assicurati che l'applicazione Node.js utilizzi la porta corretta e modifica il file `docker-compose.yml` di

in questo esempio:

- `FROM openjdk:11` : Utilizza un'immagine di base con OpenJDK 11.
- `WORKDIR /app` : Imposta la directory di lavoro nell'immagine.
- `COPY target/tuo-applicazione.jar .` : Copia il file JAR dell'applicazione (assicurati che il file JAR sia presente nella directory `target` del tuo progetto) nella directory di lavoro dell'immagine.
- `EXPOSE 8080` : Espone la porta 8080, che è la porta predefinita su cui Spring Boot di solito ascolta.
- `CMD ["java", "-jar", "tuo-applicazione.jar"]` : Specifica il comando di avvio dell'applicazione.

Puoi creare un file `docker-compose.yml` simile agli esempi precedenti per semplificare il processo di avvio del contenitore per l'applicazione Java:

```
version: '3'
services:
  java-app:
    build:
      context:
```

Autenticazione

Prima di eseguire operazioni su Docker Hub, devi autenticarti con il tuo account Docker. Puoi farlo con il comando:

```
docker login
```

Ti verrà chiesto di inserire le tue credenziali Docker Hub (nome utente, password e, se abilitata, autenticazione a due fattori).

Caricamento di un'immagine su Docker Hub

Se hai un'immagine locale che desideri caricare su Docker Hub, usa il comando `docker push`. Assicurati di taggare l'immagine con il formato `nome_utente/nome_immagine:tag` prima del push.

```
docker tag nome_immagine:tag nome_utente/nome_immagine:tag
docker push nome_utente/nome_immagine:tag
```

Scaricamento di un'immagine da Docker Hub

Per scaricare un'immagine da Docker Hub, puoi usare il comando `docker pull`. Ad esempio:

```
docker pull nome_utente/nome_immagine:tag
```

Esempi di Comandi di Base

- **Listare le immagini locali:**

```
docker images
```

- **Listare i container in esecuzione:**

```
docker ps
```

- **Listare tutti i container (inclusi quelli fermi):**

```
docker ps -a
```

- **Eliminare un'immagine locale:**

```
docker rmi nome_immagine:tag
```

- **Eliminare un container:**

```
docker rm nome_container
```

Altri Comandi Utili

- **Visualizzare informazioni sull'account Docker corrente:**

```
docker info
```

- **Eseguire comandi all'interno di un contenitore:**

```
docker exec -it nome_container comando
```

- **Visualizzare i log di un contenitore:**

```
docker logs nome_container
```

Questi sono solo esempi di comandi comuni che puoi utilizzare con Docker Hub. La CLI di Docker offre molte altre opzioni e comandi. Puoi esplorare ulteriormente la documentazione ufficiale di Docker per maggiori dettagli: Docker Command Line.