Esercitazione 2

ⓘ Differenza tra docker file e docker commit

Dockerfile e Docker commit sono due approcci diversi per creare e gestire i container Docker.

Un Dockerfile è un file di testo che contiene le istruzioni su come costruire un'immagine Docker. Di solito inizia con un'immagine di base e poi elenca una serie di istruzioni per installare le dipendenze, impostare le opzioni di configurazione e copiare i file nell'immagine. Una volta scritto il Dockerfile, è possibile utilizzare il comando docker build per creare un'immagine basata sulle istruzioni presenti nel file. Dockerfile è un approccio più strutturato e riproducibile per la creazione di immagini Docker, in quanto è possibile controllare la versione del proprio Dockerfile, tenere traccia delle modifiche e ricostruire l'immagine in modo coerente e ripetibile.

D'altra parte, Docker commit è un comando che crea una nuova immagine da un container in esecuzione. Quando si utilizza il comando docker commit, Docker prende uno snapshot dello stato corrente del container e crea una nuova immagine basata su quel punto. L'immagine risultante include tutte le modifiche apportate al container dalla sua creazione o dall'ultimo commit. Docker commit è spesso utilizzato per la sperimentazione e la prototipazione rapida, dove è possibile partire da un'immagine esistente, creare un container da essa, modificarlo e creare una nuova immagine usando docker commit.

In sintesi, Dockerfile viene utilizzato per creare un'immagine da zero, mentre Docker commit viene utilizzato per creare una nuova immagine da un container in esecuzione. Dockerfile fornisce un approccio più strutturato e controllato per la creazione di immagini Docker, mentre Docker commit è più adatto alla sperimentazione e alla prototipazione rapida.

Esempio di Dockerfile:

ϟ ATTENZIONE un Dockerfile NON deve avere estensione

Example: DockerFile

```
FROM ubuntu
RUN apt-get update && apt-get install nano
RUN echo testo > documento.txt
```

il comando da lanciare da terminale per eseguire un Dockerfile è:

```
docker build -t nomeDaDareAllImmagine:latest PercorsoDockerFile
```

se si esegue nella stessa directory basta mettere il punto al posto di "PercorsoDockerFile"

```
docker build -t nomeDaDareAllImmagine:latest .
```

Attendere il completamento della creazione dell'immagine. Durante questo processo, Docker esegue i comandi specificati nel file Dockerfile per creare l'immagine.

Verificare che l'immagine sia stata creata eseguendo il comando:

```
docker images
```

Questo comando visualizza un elenco di tutte le immagini Docker disponibili sul sistema.

Utilizzare l'immagine appena creata per eseguire un container Docker utilizzando il comando:

```
docker run -it --rm nome_immagine
```

si avvierà così un nuovo container Docker utilizzando l'immagine "nome_immagine"

Questi sono i passaggi di base per utilizzare un Dockerfile per creare un'immagine Docker. In base alla complessità del Dockerfile e alla configurazione desiderata, potrebbe essere necessario eseguire passaggi aggiuntivi o modificare il processo di creazione dell'immagine.

Tra i vari comandi disponibili troviamo ENTRIPOINT e CMD capiamo le diffirenze

ENTRYPOINT specifies the primary executable for a container, while CMD provides default arguments to that executable.

In detail:
ENTRYPOINT is used to define a default behavior for a container, which is executed when the container is run. The ENTRYPOINT instruction cannot be overridden at runtime, while CMD can be overridden.

CMD is optional, and it can be used to provide default arguments to the ENTRYPOINT or to specify a different command to run instead. If CMD is not provided, the container will run with no arguments.

ESERCIZIO

Creare un Dockerfile che permetta di fare un container con le seguenti caratteristiche:

Requirements

- **Il container deve partire da una distribuzione di ubuntu:latest**
- **Fare update e upgrade ed installare nano**
- **Esporre porta 80**
- ✓ **Verificare che il container abbia aperto la porta 80**
- ✓ **Verificare che sia installato nano**
- **Creare un file .sh che mostri il nome utente**
- ✓ **Verificare la corretta esecuzione e che il file abbia i permessi necessari**

- Creare un file contenente del testo nella macchina host (possibilmente nella stessa cartella di esecuzione del dockerfile) in attesa di essere copiato nel passo successivo
- Creare la cartella Documenti
- Copiare un file.txt dalla macchina host dentro la cartella Documenti del container
- ✓ Verificare l'esistenza della cartella e del file
- Impostare la cartella creata come default
- ✓ Verificare che all'avvio ci si ritrova nel percorso della cartella di default
- Impostare una variabile d'ambiente "mia_var" con valore 123456
- ✓ Verificare l'esistenza della nuova variabile d'ambiente
- Fare in modo che quando il container viene avviato parte direttamente da shell bash

Soluzione

```
FROM ubuntu:latest
RUN apt-get update -y
RUN apt-get upgrade -y
RUN apt-get install nano
EXPOSE 80
RUN echo whoami >> file.sh
RUN chmod +x file.sh
RUN mkdir Documenti
COPY /testo.txt /Documenti
WORKDIR /Documenti
```

```
RUN useradd -ms /bin/bash mariolino
RUN usermod -aG sudo mariolino
USER client1
ENV mia_var==123456
ENTRYPOINT ["/bin/bash"]
```