Attività 2: architettura di Docker e principali funzionalità

Esercitazione

Obiettivo: sviluppare un container Flask

Prerequisiti: Macchina Linux con Docker, sshd, gcc, Python, pip, Flask e curl installati

Fase 1: creare un'immagine basata su Ubuntu ed installare Python

1. Creare una cartella di lavoro.

2. Creare un'applicazione flask *prova.py* che restituisce un messaggio;

3. Lanciare l'applicazione, specificando - - host 0.0.0.0 in modo da mettere applicazione in ascolto su tutti gli indirizzi IP

```
$ python3 –m flask --app prova.py run --host 0.0.0.0
```

4. Provare il programma Flask utilizzando localmente il browser testuale *curl* e poi con un browser grafico da un'altra macchina in rete

5. Creare un *Dockerfile* per la generazione di un'immagine *img4* partendo da un'immagine di base Ubuntu, aggiornarla ed installare Python, pip e flask all'interno dell'immagine, copiando poi il file *prova.py* e lanciare flask (ricordarsi di cambiare i diritti di prova.py). **(per l'uso del proxy vedi la nota in fondo)**

```
FROM Ubuntu
RUN apt update
RUN apt upgrade
RUN apt –y install python3
RUN apt –y install pip
RUN python3 – m pip install flask
COPY prova.py /
CMD python3 –m flask --app prova.py run --host 0.0.0.0
```

6. Attivare il container, aprire un'altra finestra e provare con il browser testuale *curl* il funzionamento d'applicazione, sia dall'interno del container che dalla macchina host.

7. Attivare nuovamente il container, mappando una porta della macchina che ospita il container (ad esempio la porta 8080) con la porta 5000 del container (dove è in ascolto Flask)

```
$ docker container run --name cont4 --rm –it –p8080:5000 img4
```

8. Provare il container utilizzando un browser grafico da un'altra macchina in rete
9. Provare a lanciare il container in modalità detached

Uso del proxy

I comandi *RUN* all'interno del Dockerfile eseguono l'installazione di software nell'immagine in fase di generazione dell'immagine stessa.
Se in fase di generazione dell'immagine è necessario accedere ad Internet e si è in ambiente con proxy, è possibile specificare nel Dockerfile, prima del comando *RUN*, le variabili di ambiente necessarie.

```
env http_proxy=http://proxy:3128
env https_proxy=http://proxy:3128
run ……
```

Se l'immagine generata dovrà girare in un ambiente con proxy, le variabilii andranno lasciate impostate prima della riga col comando *CMD*, altrimenti andranno resettate.

```
env http_proxy=
env https_proxy=
cmd ……
```