Attività 3.b: architettura di Docker e principali funzionalità

Esercitazione

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

(accertarsi di aver aggiunto l'utente con il quale si lavora al gruppo *docker* per evitare di usare *sudo* ogni volta)

Obiettivo: creare un'immagine basata su Ubuntu (ricordarsi di impostare il proxy, se necessario)

Fase 1: creare un'immagine basata su Ubuntu

1. Creare una cartella di lavoro.

2. Creare un *Dockerfile* per la generazione di un'immagine *img3* partendo da un'immagine di base Ubuntu che esegua una *echo;* notare la differente sintassi di CMD: in questo caso viene lanciata la *shell* che esegue il comando specificato e poi chiude. Provare l'immagine

```
FROM ubuntu
CMD echo "ciao"
```

Fase 2: stati un container

1. Modificare l'immagine precedente, introducendo un ritardo di 30 secondi

```
FROM ubuntu
CMD echo "inizio"; sleep 30; echo "fine"
```

2. Eseguire il container senza l'opzione di rimozione a fine processo

```
$ docker container run --name cont3 -it img3
```

3. In una finestra *ssh* verificare lo stato del container durante e al termine dell'esecuzione

```
$ docker container ps -a
```

4. Al termine dell'esecuzione far ripartire il container

```
$ docker container start -i cont3
```

5. Rimuovere poi il container

```
$ docker container rm cont3
```

Fase 3: interagire con un container lanciato in modalità interattiva (*-it*)

1. Modificare l'immagine precedente, chiedendo il lancio di *bash*

```
FROM ubuntu
CMD ["/bin/bash"]
```

2. Eseguire il container con l'opzione di rimozione a fine processo; la console si troverà (a causa dell'opzione *–it*) associata alla *shell* del container

```
$ docker container run --name cont3 –rm –it img3
```

3. In una finestra *ssh* verificare lo stato del container: notare che il container rimane sempre attivo fino a quando si esegue *exit*

```
$ docker container ps -a
```

Fase 4: interagire con un container lanciato in modalità detached (*-dt*)

1. Eseguire il container in modalità *detached*

```
$ docker container run --name cont3 –dt img3
```

2. In una finestra *ssh* verificare lo stato del container; notare che il container rimane sempre attivo: non è possibile eseguire *exit*

```
$ docker container ps -a
```

3. In una finestra *ssh* lanciare processi all'interno del container via *exec*: ad esempio *ls -l*

```
$ docker container exec cont3 ls -l
```

4. Rimuovere poi il container, forzando l'interruzione (*-f*)

```
$ docker container rm cont3 -f
```

Fase 5: elencare i processi presenti nel container

1. Eseguire il container in modalità *detached*

```
$ docker container run --name cont3 –dt img3
```

2. Lanciare poi uno *sleep* via *exec* (questa volta la console rimarrà occupata)

```
$ docker container exec cont3 sleep 30
```

3. In una finestra *ssh* eseguire nel container il comando per elencare i processi attivi (*ps –edF*) e verificare la presenza di tre processi: il processo originario (*/bin/bash*), il processo *sleep* e lo stesso processo ps *(ps –edF)*

```
$ docker container exec cont3 ps -edF
```

4. Rimuovere poi il container, forzando l'interruzione (*-f*)

```
$ docker container rm cont3 -f
```

Fase 6: diversa visibilità dei processi all'interno e all'esterno del container

1. Eseguire il container in modalità *detached*

```
$ docker container run --name cont3 -dt img3
```

2. Lanciare poi uno *sleep* via *exec* (questa volta la console rimarrà occupata)

```
$ docker container exec cont3 sleep 30
```

3. In una finestra *ssh* eseguire nel container il comando per elencare i processi attivi (*ps –edF*) e i PID dei tre processi

```
$ docker container exec cont3 ps -edF
```

4. Elencare i processi attivi al di fuori del container e verificare che i PID assegnati sono diversi: all'interno del container è stato attivato, fra le altre cose, un *PID namespace,* causando quindi l'isolamento deiPID del container

```
$ ps -edF
```

5. Rimuovere poi il container, forzando l'interruzione (*-f*)

```
$ docker container rm cont3 -f
```