
NINJACLOUD.IT
IMMAGINI,
CONTAINER E
REGISTRI!

NINJACLOUD.IT

IF IMMAGINI

LE IMMAGINI

Cerchiamo adesso di rendere comprensibile questo concetto.

In prima analisi, possiamo identificare le **immagini come dei container stoppati**; come dei template o dei modelli da cui poter, successivamente, eseguire uno più container.

Le immagini possono essere scaricate da un REGISTRO in cui quest'ultime sono archiviate. **Il registro più conosciuto è il DOCKER HUB.** Ma non è certo l'unico!

L'OPERAZIONE DI "PULL" DI UN'IMMAGINE EFFETTUA IL DOWNLOAD DELL'IMMAGINE STESSA DA UN REGISTRO SUL NOSTRO DOCKER LOCALE E CI PERMETTE QUINDI DI ESEGUIRE IL CONTAINER DERIVATO DA TALE IMMAGINE.

NINJACLOUD.IT

Possiamo considerare l'immagine composta da più strati (layers) impilati uno di seguito all'altro e che insieme compongono un unico oggetto. Ogni strato contribuisce a rendere l'immagine completa ed utilizzabile.

IMMAGINI E STRATI

Un'immagine è composta da più strati sovrapposti. Questi strati sono in modalità "solo lettura".

Sarà compito del Docker Engine unificare e considerare tutti questi strati come una singola entità.

Per osservare questi strati all'interno dell'ambiente docker possiamo digitare il comando di pull di un'immagine già analizzato: **"docker image pull redis : lates"**.

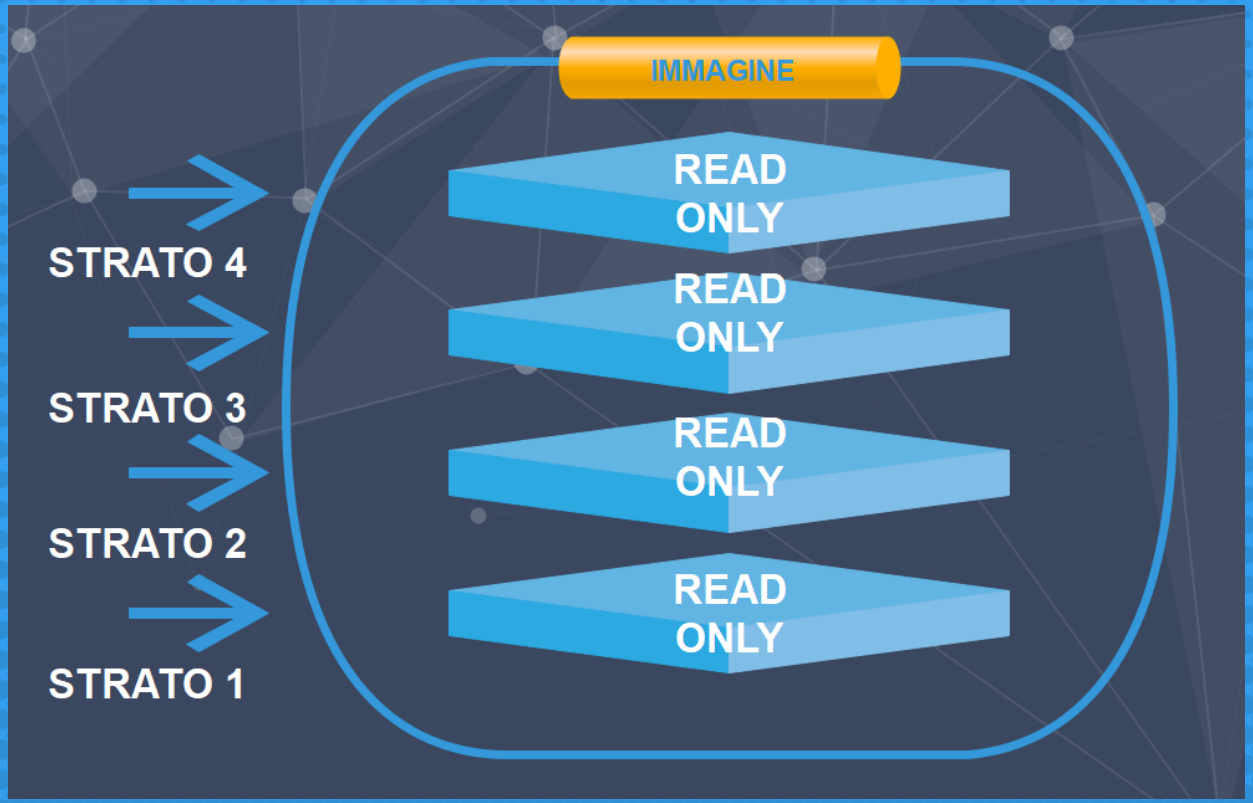


IMMAGINE
READ ONLY
READ ONLY
READ ONLY
READ ONLY
STRATO 4
STRATO 3
STRATO 2
STRATO 1

NINJACLOUD.IT

ARGOMENTO RICHIESTO AI FINI DELLA CERTIFICAZIONE!!

ESTRATTO DALLA STUDY GUIDE:

"Summarize how an application is composed of layers and where those layers reside on the filesystem".

OSSERVIAMO ADESSO I VARI STRATI CHE COMPONGONO UN'IMMAGINE TRAMITE IL COMANDO "DOCKER IMAGE PULL...".

NINJACLOUD.IT

Un'altra modalità per osservare gli strati di un'immagine è tramite il comando "docker inspect redis:latest".

NINJACLOUD.IT

ARGOMENTO RICHIESTO AI FINI DELLA CERTIFICAZIONE!!

ESTRATTO DALLA STUDY GUIDE:

“Display layers of a Docker image”.

OSSERVIAMO ADESSO I VARI STRATI CHE COMPONGONO UN'IMMAGINE TRAMITE IL COMANDO "DOCKER INSPECT...".

NINJACLOUD.IT

ARGOMENTO RICHIESTO AI FINI DELLA CERTIFICAZIONE!!

ESTRATTO DALLA STUDY GUIDE:

“Interpret the output of "docker inspect" commands”.

IMMAGINI E STRATI

Notare che **tutte le immagini Docker nascono con uno STRATO DI BASE**. Ogni **modifica** effettuata all'immagine **incrementerà** il **numero di strati**. Successivamente quando analizzeremo il dockerfile vedremo come i comandi contenuti in quest'ultimo contribuiscono all'incremento degli strati che compongono l'immagine.

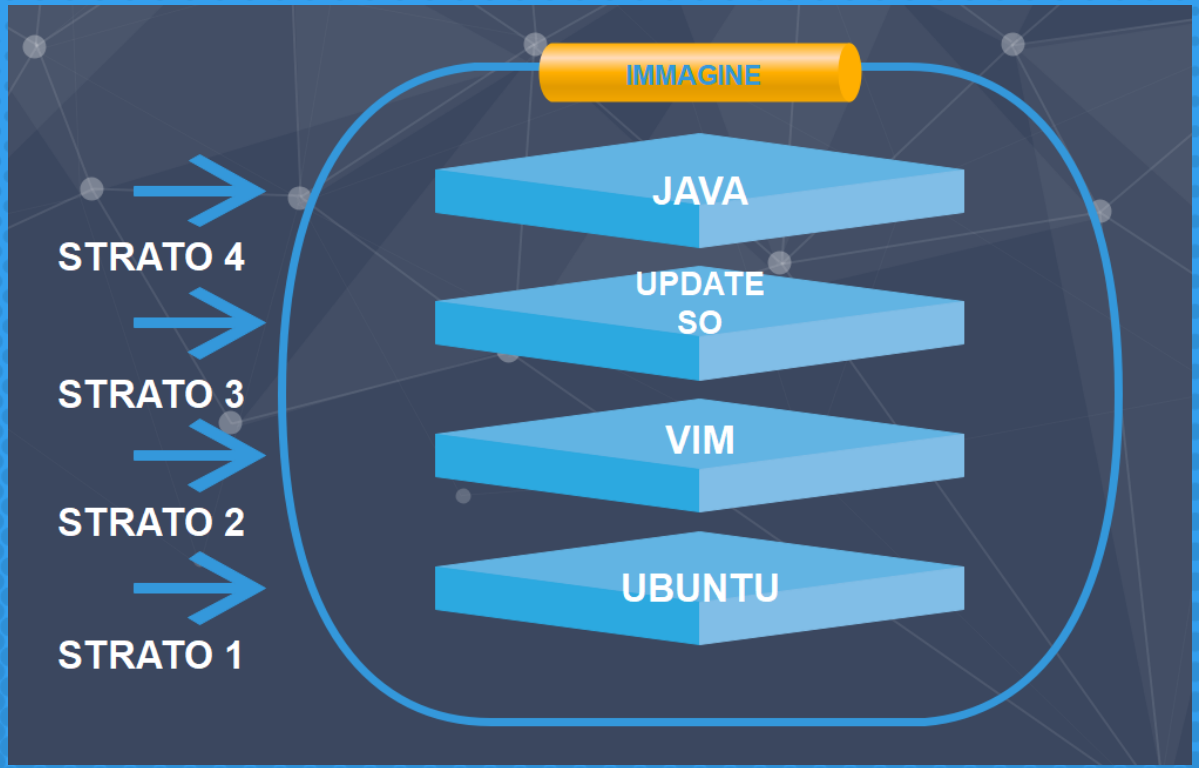


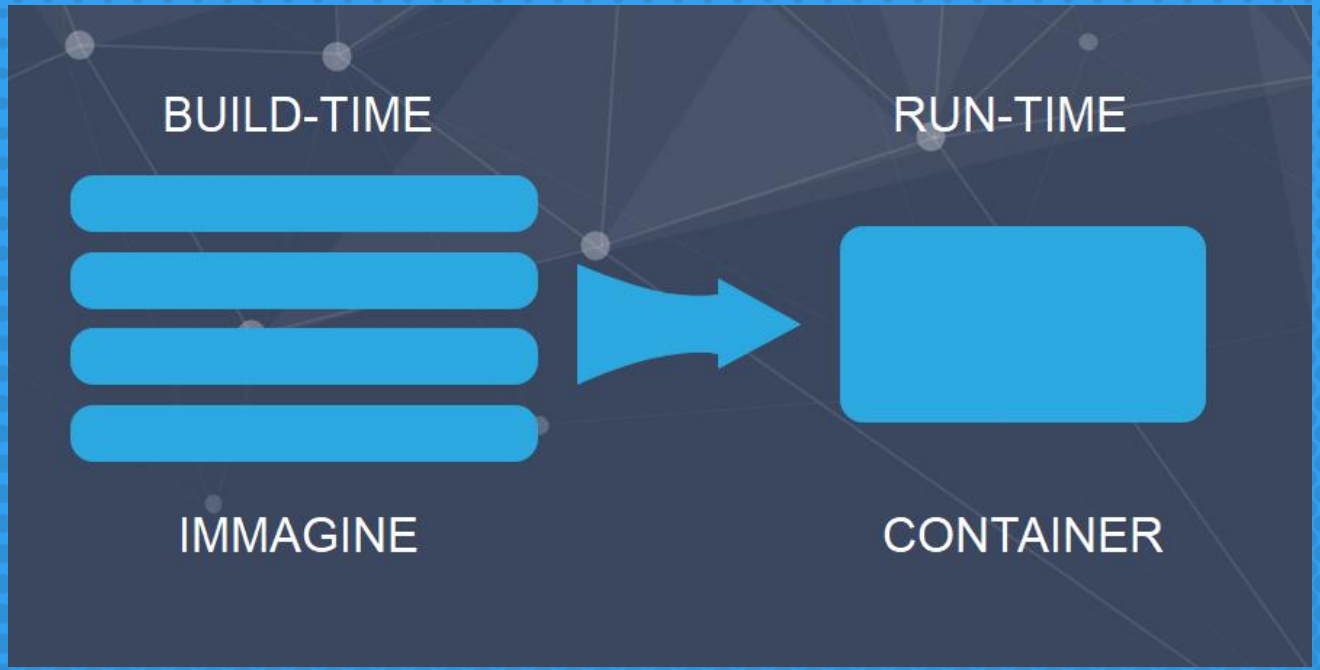
IMMAGINE
JAVA
UPDATE
SO
VIM
UBUNTU
STRATO 4
STRATO 3
STRATO 2
STRATO 1

NINJACLOUD.IT

L'IMMAGINE AVRÀ IL SOLO SCOPO DI PERMETTERE L'ESECUZIONE DI UN'APPLICAZIONE O SERVIZIO. PER QUESTO MOTIVO SARÀ SOLITAMENTE LEGGERA E FORNITA DELLE COMPONENTI ESSENZIALI.

BUILD-TIME VS RUN-TIME

Possiamo definire le immagini come delle entità **“build-time”** ovvero che assumono significato solo nel momento della loro definizione a differenza dei container che sono entità **“run-time”** ovvero che assumono significato solo durante la loro esecuzione. Un’immagine sarà sempre un elemento statico, immobile da cui poter derivare molteplici istanze (container).


BUILD-TIME
RUN-TIME
IMMAGINE
CONTAINER

NINJACLOUD.IT

Fare attenzione a non confondere il concetto di immagine con quello di container. Questo è il primo passo verso la comprensione dell'ecosistema Docker.

NINJACLOUD.IT

TENERE CONTO CHE SI CREA UNA DIPENDENZA TRA IMMAGINE E CONTAINER. SE ESEGUIAMO UN CONTAINER, QUEST'ULTIMO DIPENDE DALL'IMMAGINE DA CUI È DERIVATO, INFATTI **NON È POSSIBILE ELIMINARE UN'IMMAGINE FINCHÈ QUESTA È IN USO DAL CONTAINER**.

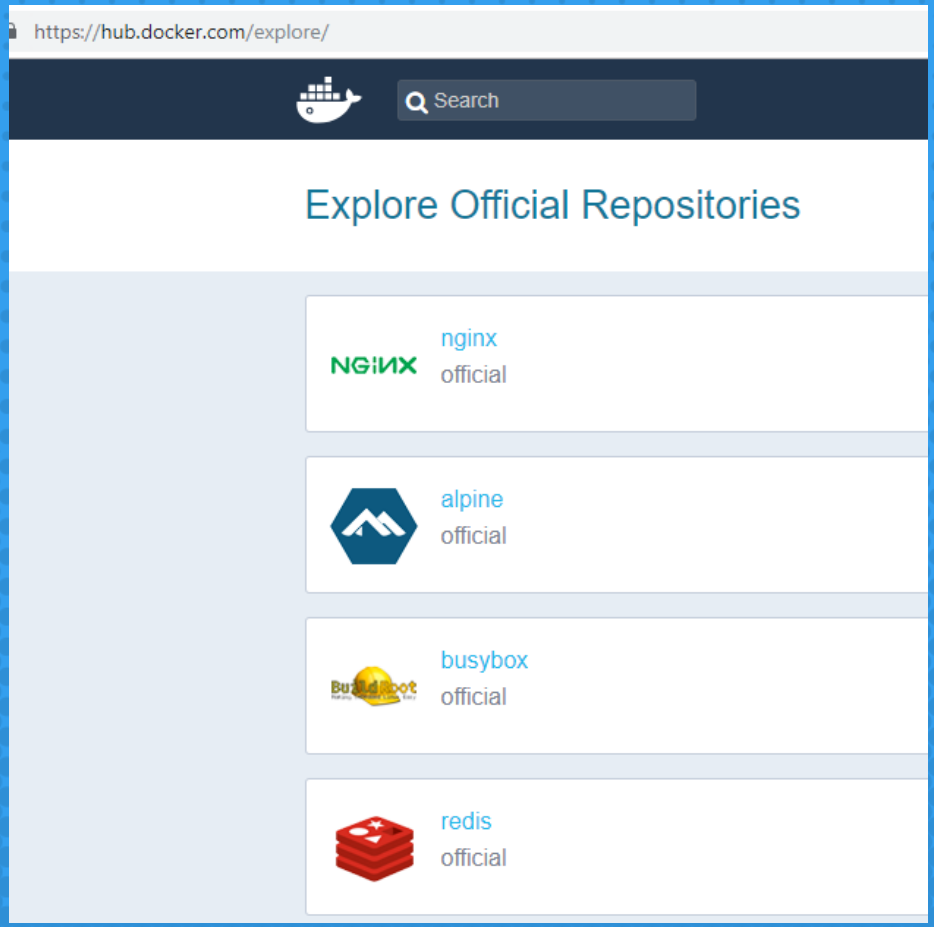
DOCKER HUB

Avremo modo di approfondire il Docker Hub più volte durante questo corso, tuttavia, dal momento che stiamo analizzando il concetto di immagine e il **Docker Hub è il "raccoglitore" delle immagini e delle loro versioni**, ha senso prendere confidenza con quest'ultimo.

NINJACLOUD.IT

DOCKER HUB

Il Docker Hub ci permette di selezionare l'immagine che più si adatta alle nostre esigenze e le immagini al suo interno sono migliaia. È fondamentale prendere confidenza con quest'ultimo in quanto **la corretta scelta dell'immagine può far la differenza** e migliorare l'esito di un progetto.


https://hub.docker.com/explore/
Search
Explore Official Repositories
nginx
official
alpine
official
busybox
official
redis
official

I REGISTRI

Esistono anche altri registri, oltre al Docker Hub, alcuni di questi sono locati on-premise così da garantire un buon livello di sicurezza.

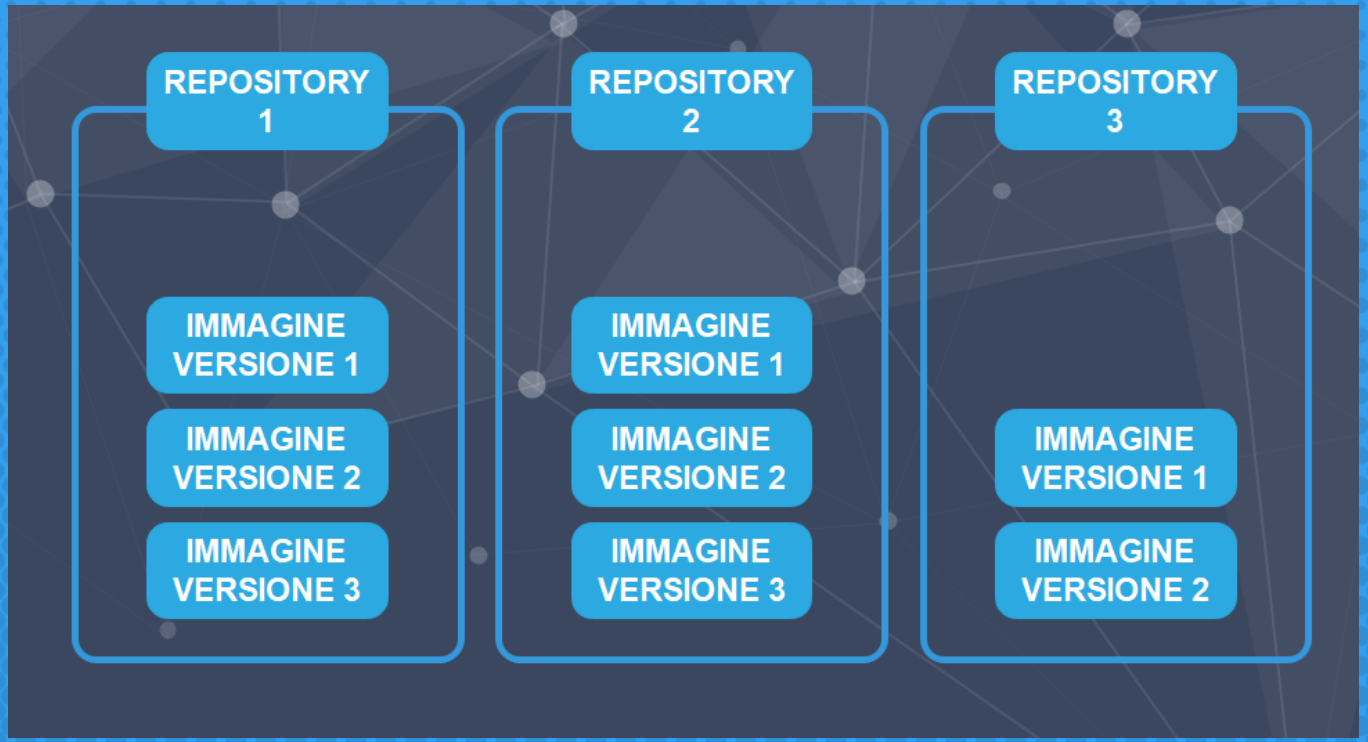
I registri contengono uno o più **REPOSITORY**. Un repository contiene una o più **IMMAGINI**.

I REGISTRI

Un'ulteriore distinzione, presente anche sul Docker HUb, è tra **REPOSITORY UFFICIALE** e **NON UFFICIALE**.

I repository ufficiali contengono spesso delle immagini qualitativamente migliori, più aggiornate e sicure.

La maggior parte dei sistemi operativi e delle applicazioni hanno il loro repository ufficiale sul Docker Hub.

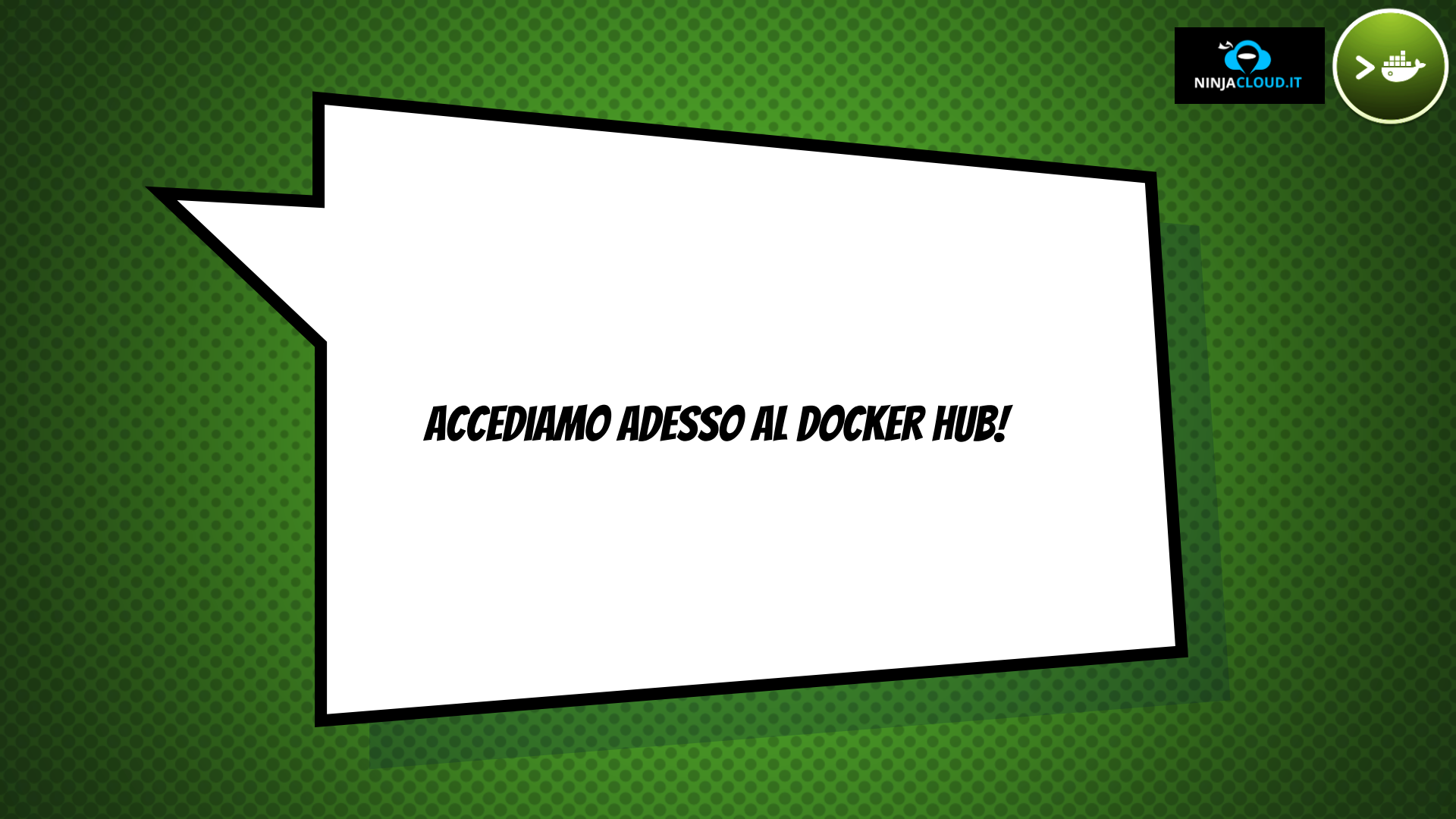

REPOSITORY 1
IMMAGINE VERSIONE 1
IMMAGINE VERSIONE 2
IMMAGINE VERSIONE 3
REPOSITORY 2
IMMAGINE VERSIONE 1
IMMAGINE VERSIONE 2
IMMAGINE VERSIONE 3
REPOSITORY 3
IMMAGINE VERSIONE 1
IMMAGINE VERSIONE 2

NINJACLOUD.IT

ARGOMENTO RICHIESTO AI FINI DELLA CERTIFICAZIONE!!

ESTRATTO DALLA STUDY GUIDE:

“Utilize a registry to store an image”.


NINJACLOUD.IT
ACCEDIAMO ADESSO AL DOCKER HUB!

NINJACLOUD.IT

UNA NUOVA INSTALLAZIONE DI DOCKER NON HA NESSUNA IMMAGINE DISPONIBILE LOCALMENTE. È POSSIBILE VERIFICARLO DIGITANDO IL COMANDO: "DOCKER IMAGE LS".

"PULLING" DI UN'IMMAGINE

Come già anticipato, effettuare il **"pull"** di un'immagine significa **scaricare l'immagine stessa sul docker installato localmente**. E' possibile specificare anche la versione ma lo vedremo più avanti. Docker verifica sempre se nella cache locale è già presente l'immagine o meno. Se non è presente procede con l'operazione di "pulling".

IL COMANDO PER EFFETTUARE IL "PULLING" DI UN'IMMAGINE DAL REPOSITORY UFFICIALE È:

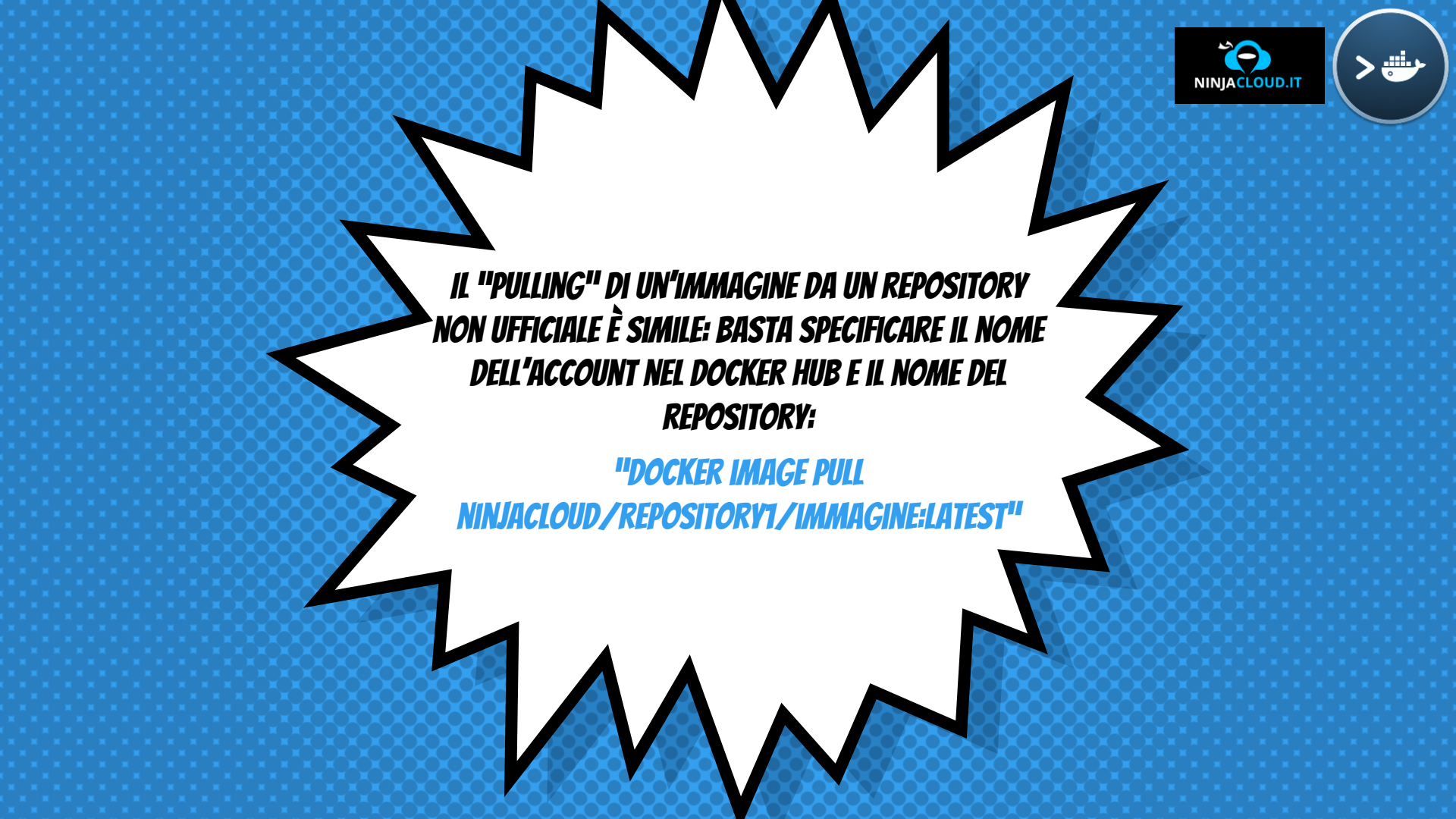
"DOCKER IMAGE PULL <NOME_REPOSITORY> : <TAG_IMMAGINE>"

ESEMPIO:

"DOCKER IMAGE PULL REDIS : 5.0.1" -> VERRÀ SCARICATA L'IMMAGINE DOCKER RELATIVA ALLA VERSIONE DI REDIS 5.0.1
"DOCKER IMAGE PULL REDIS : LATEST" -> VERRÀ SCARICATA L'IMMAGINE DOCKER RELATIVA ALL'ULTIMA VERSIONE DI REDIS.
"DOCKER IMAGE PULL REDIS" -> SE NON SPECIFICHIAMO NESSUN TAG VERRÀ SCARICATA L'ULTIMA VERSIONE COME SPECIFICANDO IL TAG "LATEST".

NINJACLOUD.IT

Il comando "docker run..." si utilizza per l'avvio di un container. Tuttavia possiamo utilizzarlo anche per effettuare il pull di un'immagine: prima verificherà se l'immagine è gia presente localmente, poi provvederà allo scaricamento e all'esecuzione del container.


NINJACLOUD.IT
IL "PULLING" DI UN'IMMAGINE DA UN REPOSITORY NON UFFICIALE È SIMILE: BASTA SPECIFICARE IL NOME DELL'ACCOUNT NEL DOCKER HUB E IL NOME DEL REPOSITORY:
"DOCKER IMAGE PULL NINJACLOUD/REPOSITORY1/IMMAGINE:LATEST"

NOTA: IL TAG "LATEST" NON GARANTISCE TUTTAVIA CHE LA VERSIONE DELL'IMMAGINE SCARICATA SIA EFFETTIVAMENTE L'ULTIMA. E' SEMPLICEMENTE UN'ETICHETTA. DOBBIAMO SEMPRE VERIFICARE CHE SIA IN EFFETTI COSÌ.

NINJACLOUD.IT

ARGOMENTO RICHIESTO AI FINI DELLA CERTIFICAZIONE!!

ESTRATTO DALLA STUDY GUIDE:

“Demonstrate tagging an image”.

NINJACLOUD.IT

EFFETTUIAMO ADESSO IL "PULLING" DI QUALCHE IMMAGINE DIRETTAMENTE DAL DOCKER HUB.

NINJACLOUD.IT

"PULLING" DI UN'IMMAGINE TRAMITE IL "DIGEST"

Solitamente il "pulling" di un'immagine viene effettuato utilizzando il meccanismo dei tag. Tuttavia per assicurarci che l'immagine scaricata sia effettivamente quella voluta e che i tag siano corretti, si può **utilizzare un "hash" chiamato "digest". La particolarità e il punto di forza del "digest" è che il suo valore è IMMUTABILE.**

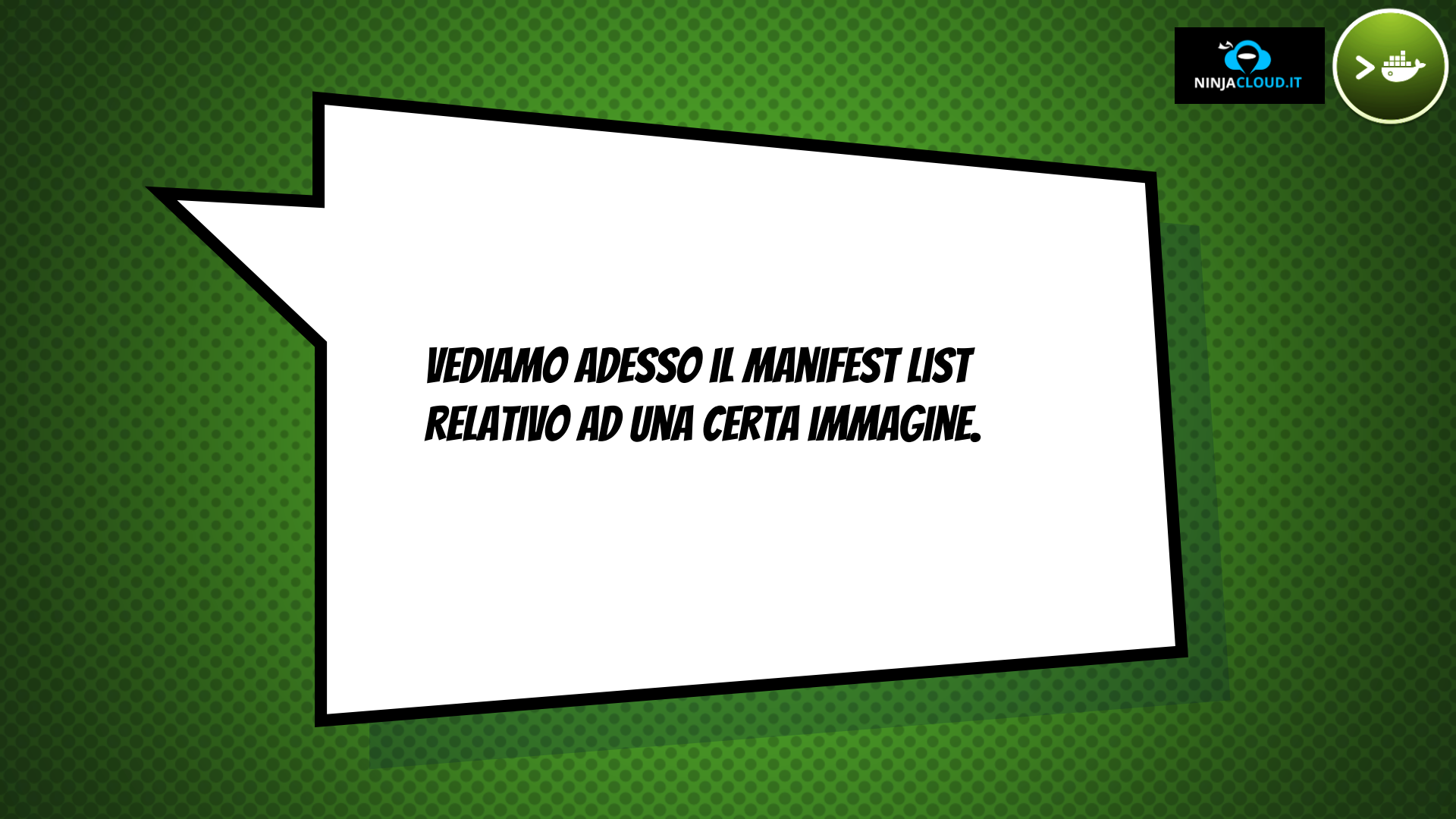
Ogni volta che effettuiamo il "pull" di un'immagine ci viene fornito in output anche il digest di tale immagine. E' sufficiente copiarlo e conservarlo, nel caso di un nuovo "pull" potremmo confrontare i valori e se saranno diversi significherà che è stato apportato un cambiamento all'immagine.

VEDIAMO ADESSO IL "PULLING" DI UN'IMMAGINE TRAMITE IL "DIGEST"

MANIFEST LIST E IMMAGINI MULTI ARCHITETTURA

Dal momento che la stessa immagine e la stessa versione, può essere adatta al "pulling" da parte di architettura differenti, si è introdotto il concetto di "manifest list" che contiene appunto la lista delle architetture disponibili relativamente ad una certa immagine e versione (tag).

Non dobbiamo specificare noi l'architettura, Docker lo farà in automatico.


NINJACLOUD.IT
VEDIAMO ADESSO IL MANIFEST LIST
RELATIVO AD UNA CERTA IMMAGINE.


NINJACLOUD.IT
DI SEGUITO ALCUNI DEI COMANDI PIÙ UTILI PER QUANTO RIGUARDA LA GESTIONE DELLE IMMAGINI!

1.

"DOCKER IMAGE LS"

Per visualizzare la lista delle immagini presenti nel nostro ambiente Docker locale

NINJACLOUD.IT

2.

"DOCKER IMAGE" OPPURE "DOCKER IMAGES --HELP"

Per visualizzare tutte le possibilità d'uso del comando.

NINJACLOUD.IT

3.

"DOCKER IMAGE PULL" OPPURE "DOCKER PULL"

Per effettuare il pulling di un'immagine da un REGISTRO. L'abbiamo già esaminato.

NINJACLOUD.IT

4.

"DOCKER IMAGES -Q"

Restituisce solo l'ID numerico dell'immagine.

NINJACLOUD.IT

5.

"DOCKER IMAGES -F"

L'opzione "–f" permette di inserire un filtro e selezionare solo le immagini richieste.

NINJACLOUD.IT

6.

"DOCKER RMI" OPPURE "DOCKER IMAGE RM"

Per rimuovere un'immagine dall'ambiente Docker locale.

NINJACLOUD.IT

7.

"DOCKER INSPECT NOME_IMMAGINE:TAG"

Per avere delle informazioni dettagliate su una certa immagine Docker.

NINJACLOUD.IT

8.

"DOCKER SEARCH STRINGA"

Ricerca all'interno del Docker Hub tutti i repository che contengono la stringa specificata nel comando.

NINJACLOUD.IT

9.

"DOCKER RUN NOME_IMMAGINE"

Permette di eseguire un container a partire da una certa immagine specificata. Vedremo nel dettaglio successivamente.

NINJACLOUD.IT

10.

"DOCKER HISTORY NOME_IMMAGINE"

Permette di visualizzare lo storico delle operazioni svolte relativamente ad un determinata immagine.

NINJACLOUD.IT

ARGOMENTO RICHIESTO AI FINI DELLA CERTIFICAZIONE!!

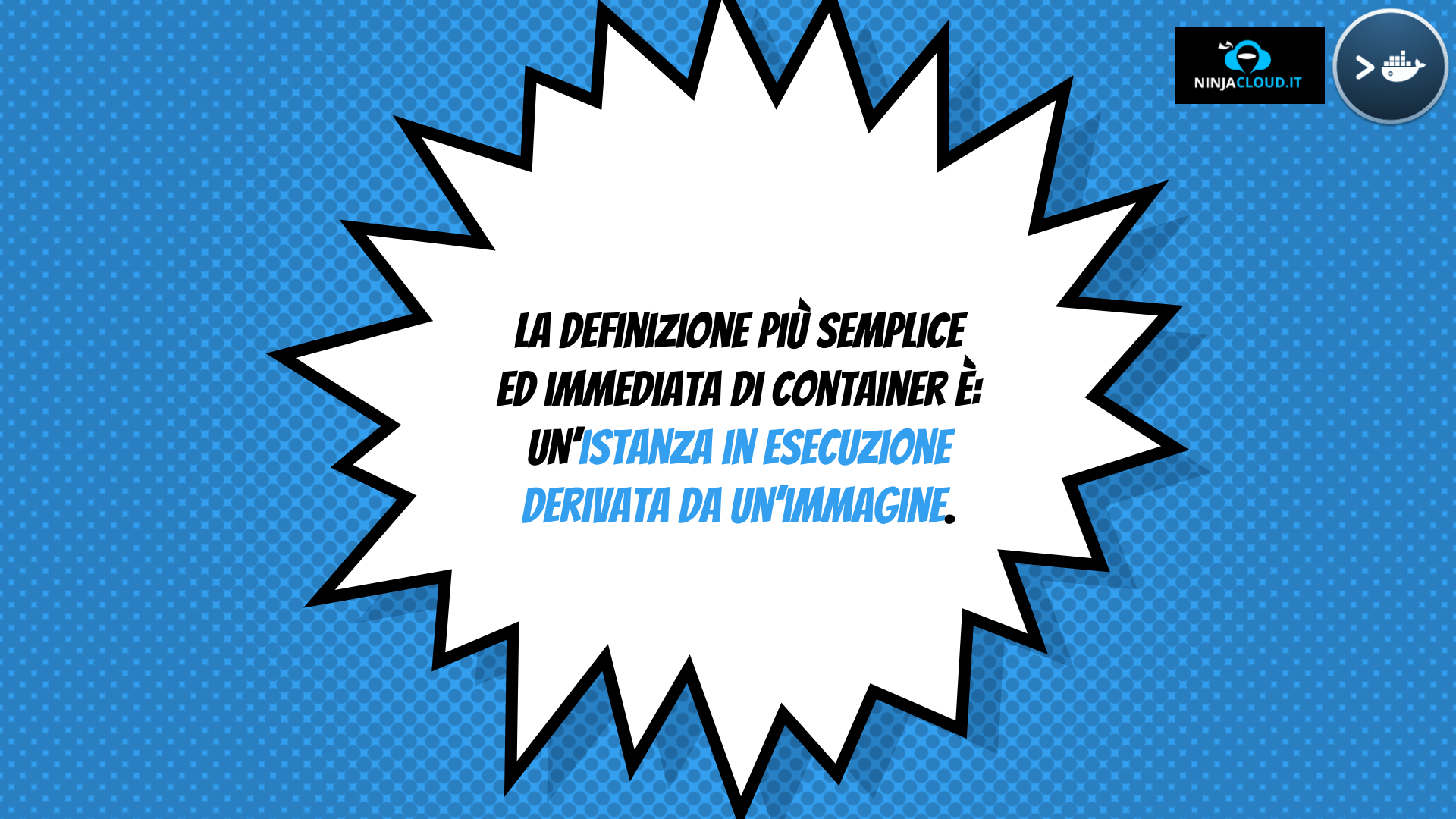
ESTRATTO DALLA STUDY GUIDE:

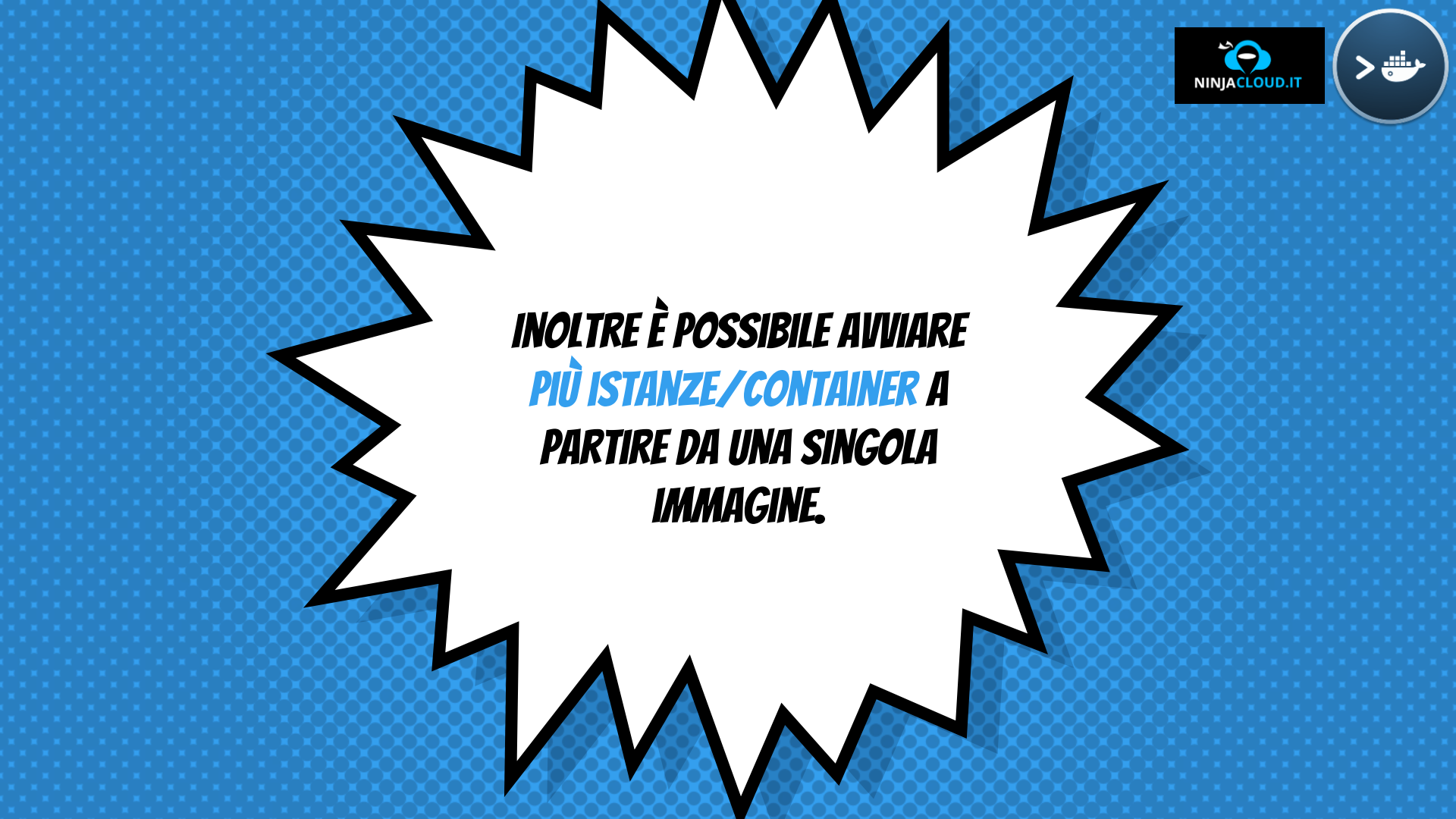
"Use CLI commands such as list, delete, prune, rmi, etc to manage images".

TESTIAMO ADESSO I COMANDI ELENCATI IN PRECEDENZA.

NINJACLOUD.IT

CONTAINER


NINJACLOUD.IT
LA DEFINIZIONE PIÙ SEMPLICE
ED IMMEDIATA DI CONTAINER È:
UN'ISTANZA IN ESECUZIONE
DERIVATA DA UN'IMMAGINE.


NINJACLOUD.IT
INOLTRE È POSSIBILE AVVIARE
PIÙ ISTANZE/CONTAINER A
PARTIRE DA UNA SINGOLA
IMMAGINE.

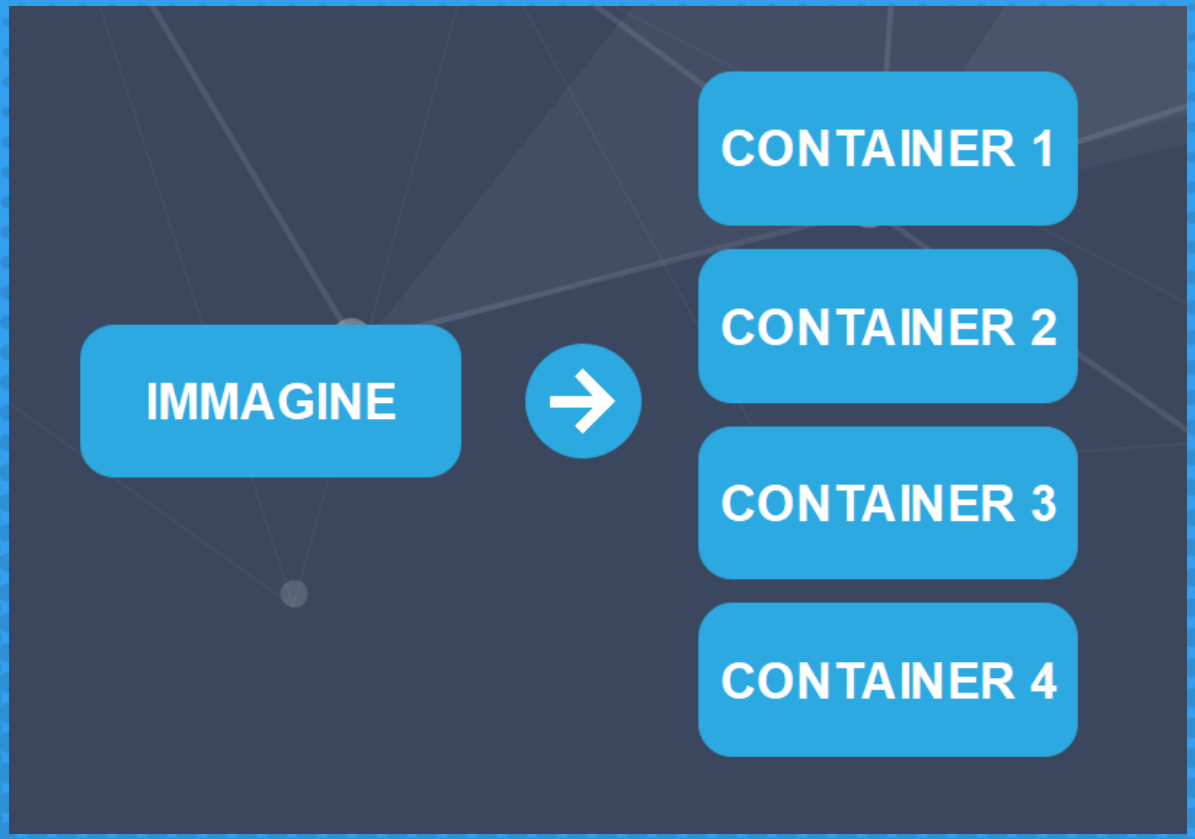


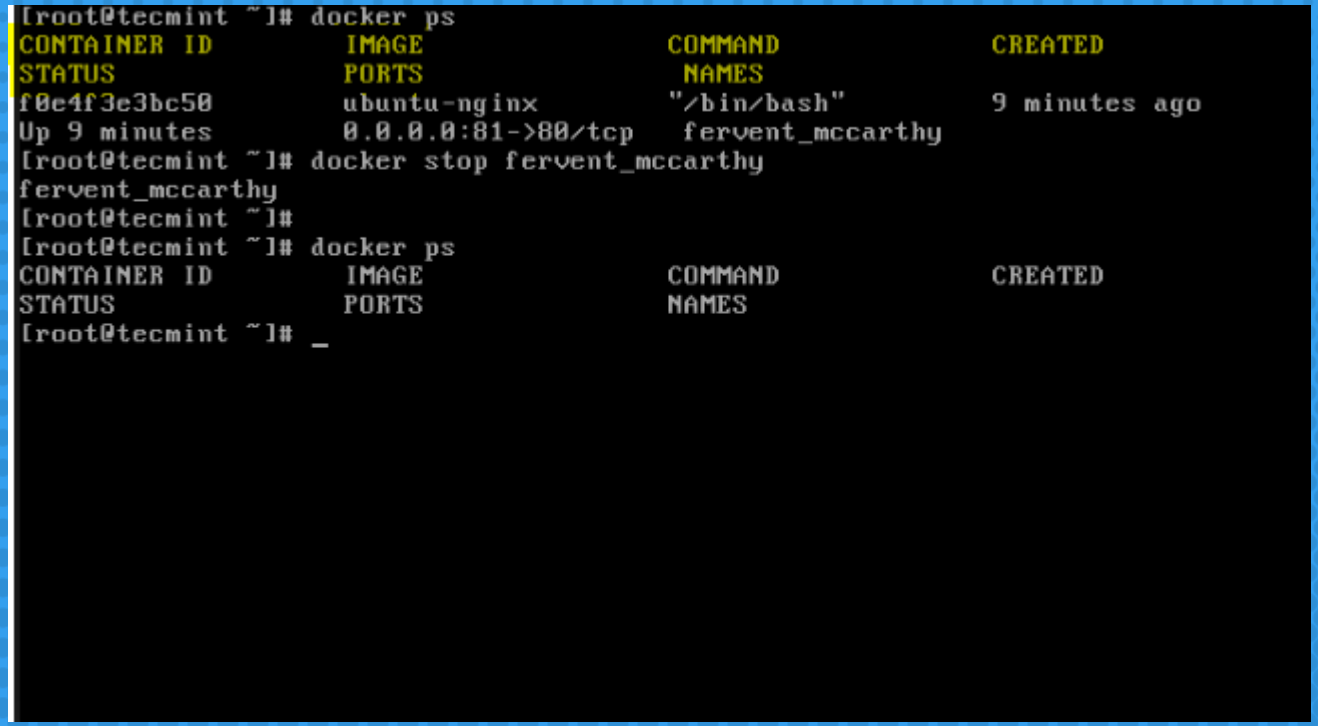
IMMAGINE
CONTAINER 1
CONTAINER 2
CONTAINER 3
CONTAINER 4

AVVIARE UN CONTAINER

Il comando **"docker container run"** ci permette di eseguire un container. Questo è il comando nella sua forma base, come vedremo abbiamo la possibilità di aggiungere delle opzioni e parametri.

Normalmente si utilizza un container allo scopo di utilizzare un'applicazione o un servizio; il comando assume quindi la seguente forma: **"docker container run <image> <app>"**.

Ad esempio: **"docker container run –it ubuntu /bin/bash"** ci permette di avviare una shell di tipo "bash" all'interno del container Ubuntu.


[root@tecmint ~]# docker ps
CONTAINER ID IMAGE COMMAND CREATED
STATUS PORTS NAMES
f0e4f3e3bc50 ubuntu-nginx "/bin/bash" 9 minutes ago
Up 9 minutes 0.0.0.0:81->80/tcp fervent_mccarthy
[root@tecmint ~]# docker stop fervent_mccarthy
fervent_mccarthy
[root@tecmint ~]#
[root@tecmint ~]# docker ps
CONTAINER ID IMAGE COMMAND CREATED
STATUS PORTS NAMES
[root@tecmint ~]# _

NINJACLOUD.IT

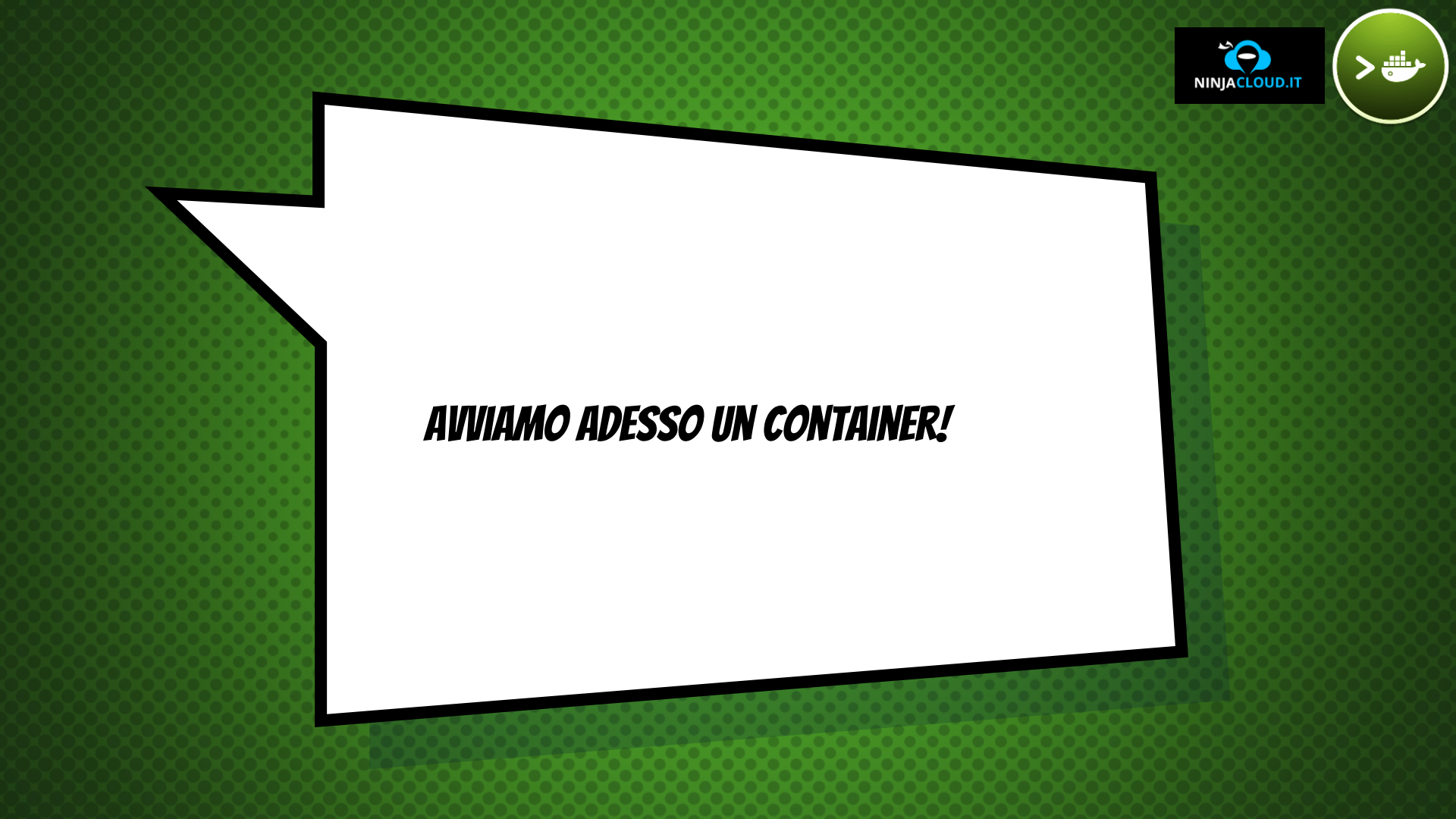
AVVIARE UN CONTAINER

L'opzione **"-it"** collega il nostro terminale a quello del container così da poter effettuare le attività all'interno di quest'ultimo.

Nota che un container resta in esecuzione finchè l'applicazione o il servizio in esecuzione esiste.

Il comando **"spleep"** ci permette di effettuare un test proprio in tal senso.

Inoltre, avviando un container ci rendiamo presto conto che alcuni comandi che normalmente utilizziamo non sono disponibili: questo è normale. L'immagine contiene solo ciò che è strettamente necessario. Ovviamente abbiamo la possibilità di scaricare in seguito ciò che ci occorre.


NINJACLOUD.IT
AVVIAMO ADESSO UN CONTAINER!

ANALISI DI UN CONTAINER

Con il comando **"ps –elf"** possiamo verificare i processi in esecuzione all'interno del container.

Ripetiamo ancora una volta: il container esiste solo in funzione dell'applicazione o servizio attualmente in esecuzione. Per cui se stoppiamo il processo associato a tale applicazione, anche il container cesserà la sua esecuzione.

Per uscire da un container in esecuzione, possiamo digitare la combinazione di tasti: **"ctrl+P e ctrl+Q"** così facendo il container resterà in esecuzione e noi torneremo sul terminal dell'host.

Per rientrare sul terminale del container digitiamo il comando: **"docker container exec –it ID_CONTAINER bash"**.

Per verificare se il container è effettivamente in esecuzione digitare il comando: "docker container ls".

NINJACLOUD.IT

Con il comando "docker ps" possiamo verificare i container in esecuzione.
Con il comando "docker ps -a" anche i container stoppati.

NINJACLOUD.IT

PER EFFETTUARE LO START DI UN CONTAINER STOPPATO SI UTILIZZA IL COMANDO "DOCKER CONTAINER START ID_CONTAINER".

NINJACLOUD.IT

PER STOPPARE UN CONTAINER SI UTILIZZA IL COMANDO: "DOCKER CONTAINER STOP ID_CONTAINER". PER RIMUOVERLO: "DOCKER RM ID_CONTAINER".


NINJACLOUD.IT
Il comando "docker container prune" rimuove tutti i container che sono stati stoppati.

NINJACLOUD.IT

È FORTEMENTE CONSIGLIATO PRIMA DI RIMUOVERE UN CONTAINER DI EFFETTUARE LO STOP DI QUEST'ULTIMO. QUESTO GARANTISCE UNA CORRETTA GESTIONE DEI PROCESSI E QUINDI UNA SITUAZIONE PIÙ "PULITA".

NINJACLOUD.IT

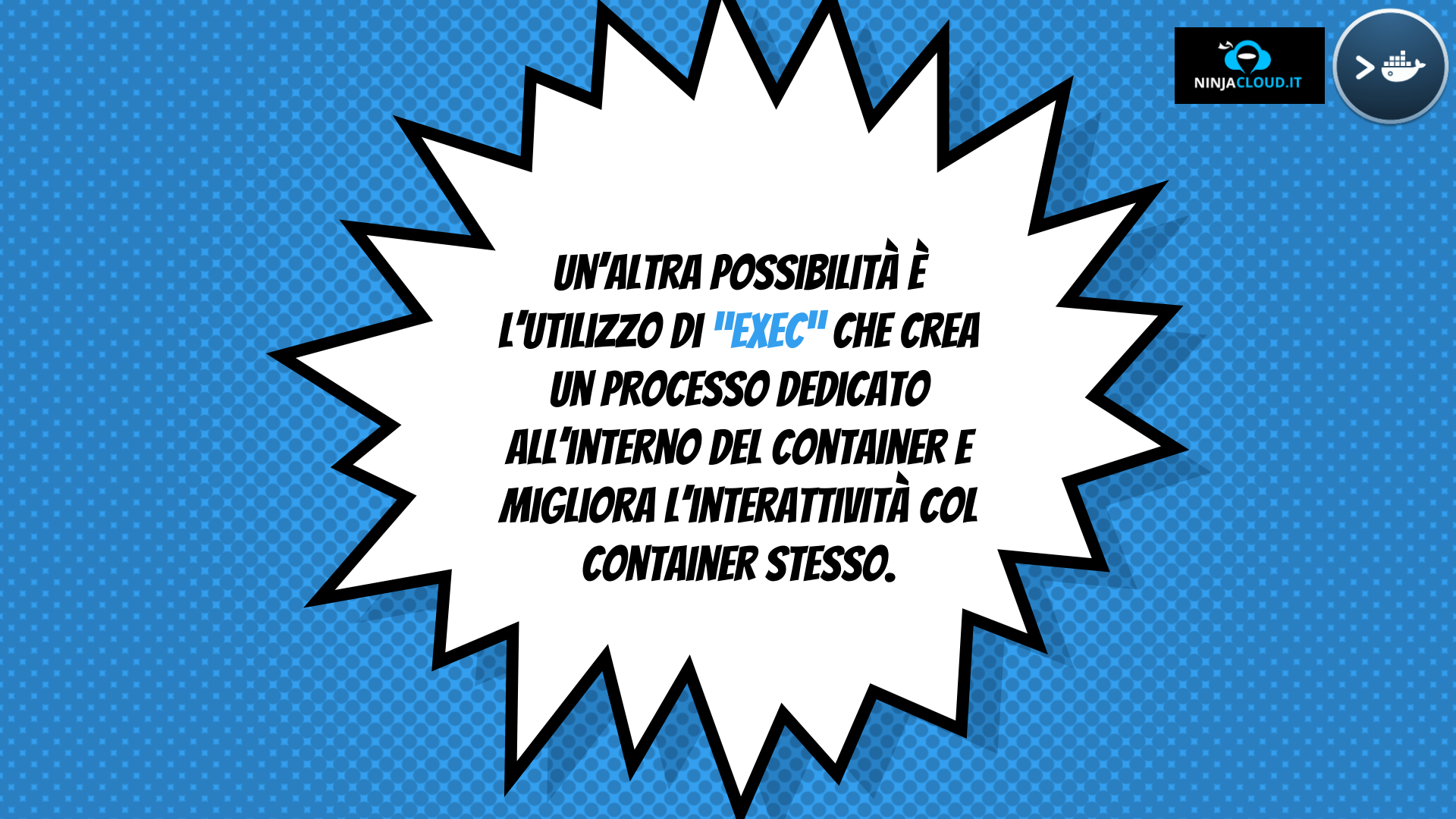
PER AVVIARE UN CONTAINER IN BACKGROUND SI UTILIZZA IL FLAG "-D".

NINJACLOUD.IT

IL FLAG "--NAME" CI PERMETTE
DI DARE UN NOME AL
CONTAINER IN ESECUZIONE.

Altri due comandi utili sono "docker top id_container" e "docker stats id_container".

Per accedere ad un container possiamo utilizzare anche il comando "docker attach ID_CONTAINER".


NINJACLOUD.IT
UN'ALTRA POSSIBILITÀ È L'UTILIZZO DI "EXEC" CHE CREA UN PROCESSO DEDICATO ALL'INTERNO DEL CONTAINER E MIGLIORA L'INTERATTIVITÀ COL CONTAINER STESSO.

NINJACLOUD.IT

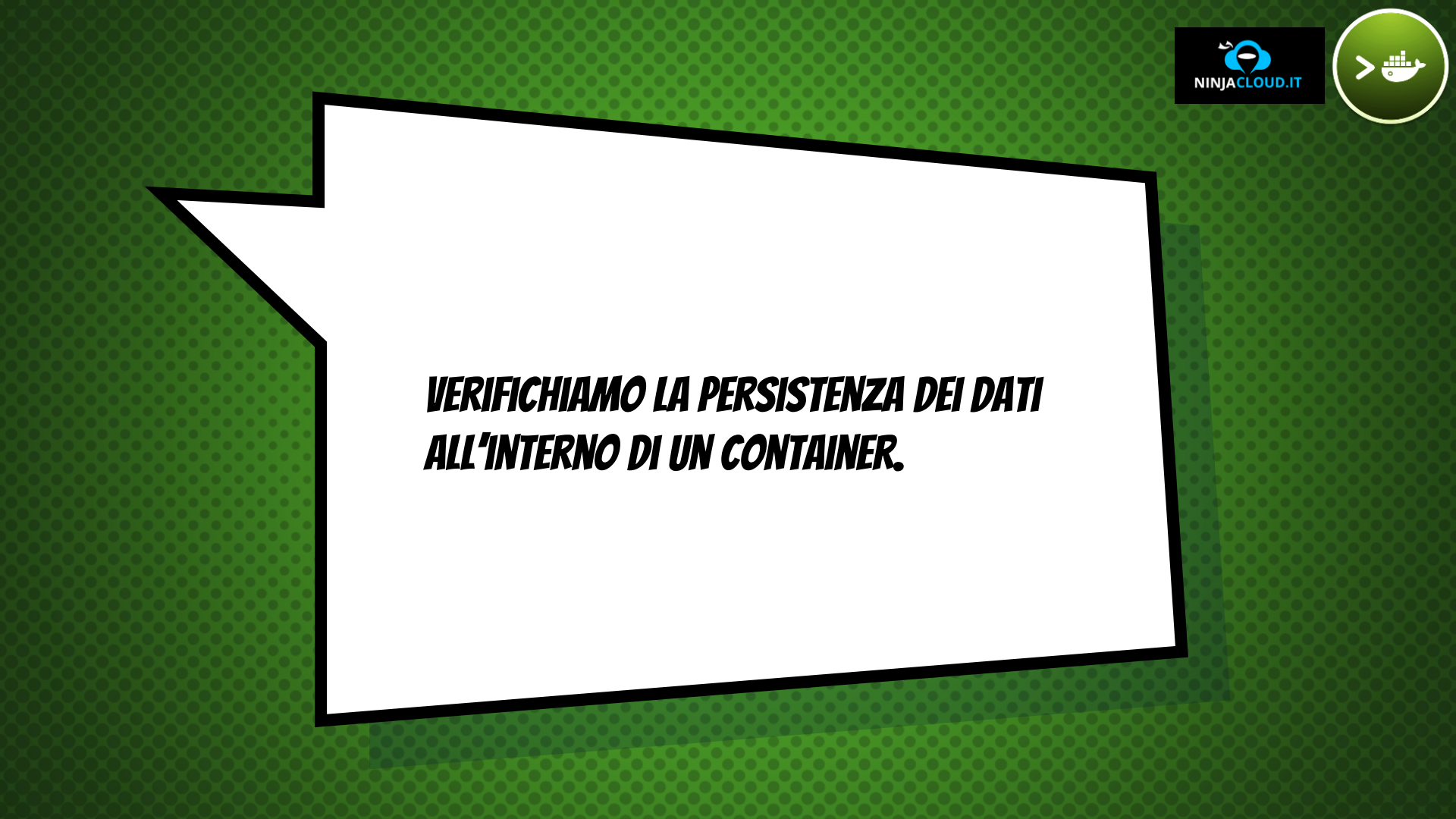
PER VISUALIZZARE I LOG DI UN CONTAINER POSSIAMO UTILIZZARE IL COMANDO: "DOCKER CONTAINER LOGS ID_CONTAINER".

NINJACLOUD.IT

TESTIAMO I COMANDI IN LABORATORIO!

NINJACLOUD.IT

Notare che le informazioni all'interno di un container persistono finchè quest'ultimo non viene eliminato definitivamente. Lo stop di un container non porta alla perdita dei dati al suo interno. Nella sezione dedicata ai volumi vedremo come andare addirittura oltre.


NINJACLOUD.IT
VERIFICHIAMO LA PERSISTENZA DEI DATI
ALL'INTERNO DI UN CONTAINER.

POLICY DI RESTART

È una best practice consigliata quella di creare una RESTART POLICY che permette il riavvio automatico del container al verificarsi di un qualche tipo di evento, imprevisto o meno che sia.

Esistono tre tipologie di restart policy:

- **ALWAYS.**
- **UNLESS-STOPPED.**
- **ON-FAILED.**

NINJACLOUD.IT

POLICY DI RESTART

La prima tipologia **(ALWAYS)** riavvia sempre il container salvo se quest'ultimo è stato manualmente stoppato. Se viene riavviato il Docker Daemon tutti i container che sono stoppati e che contengono la policy "restart always" saranno avviati.

La seconda tipologia **(UNLESS-STOPPED)** non riavvia il container anche se quest'ultimo è stato stoppato e il Docker Daemon si è riavviato.

La terza tipologia **(ON-FAILED)** riavvia il container se quest'ultimo esce con un "non-zero exit code" e lo riavvia nel caso il Docker Daemon sia riavviato e lo stato sia "stopped".

NINJACLOUD.IT

VEDIAMO QUALCHE ESEMPIO PRATICO PER CHIARIRE MEGLIO IL CONCETTO.

ESEGUIAMO ADESSO UN SEMPLICE CONTAINER CHE CONTIENE UN WEB SERVER.

NINJACLOUD.IT

PORT MAPPING

Il **"port mapping"** è un argomento che tratteremo nel dettaglio nel seguito di questo corso.

Giusto una spiegazione preliminare: **il comando "-p 80:8080" associa la porta del container con la porta dell'host sul quale stiamo eseguendo Docker.**

Nell'esempio stiamo mappando la porta 80 dell host con la porta 8080 del container. L'associazione avviene sempre secondo questa logica: porta dell'host : porta del container.

Significa che tutto il traffico che arriverà sulla porta 80 dell'host sarà rediretto sulla porta 8080 del container. **Questo è un argomento fondamentale che verrà ripreso nel dettaglio successivamente.**