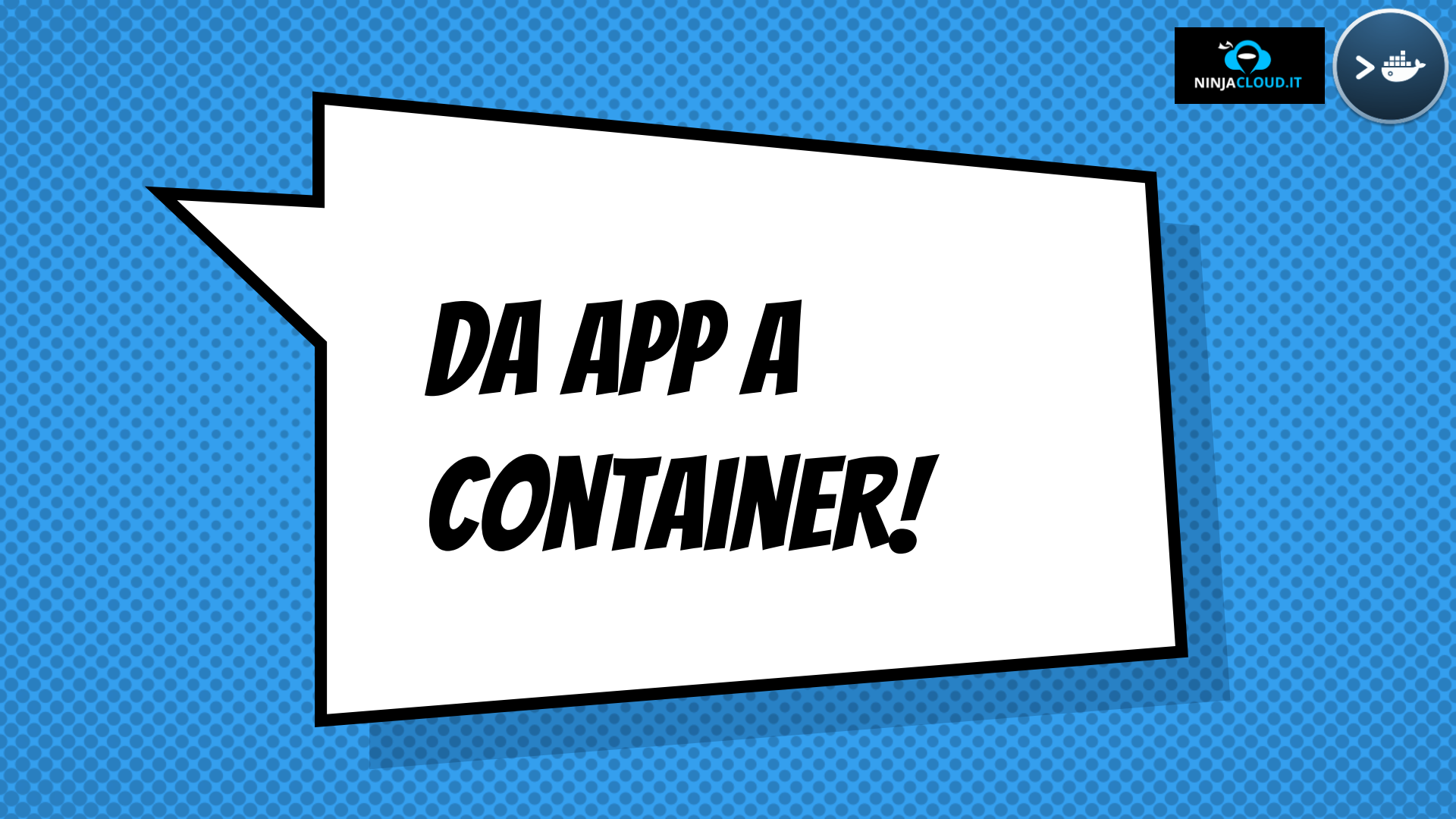
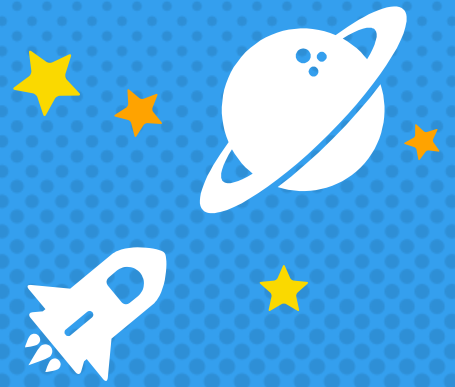

NINJACLOUD.IT
DA APP A CONTAINER!

NINJACLOUD.IT

DA APP A CONTAINER

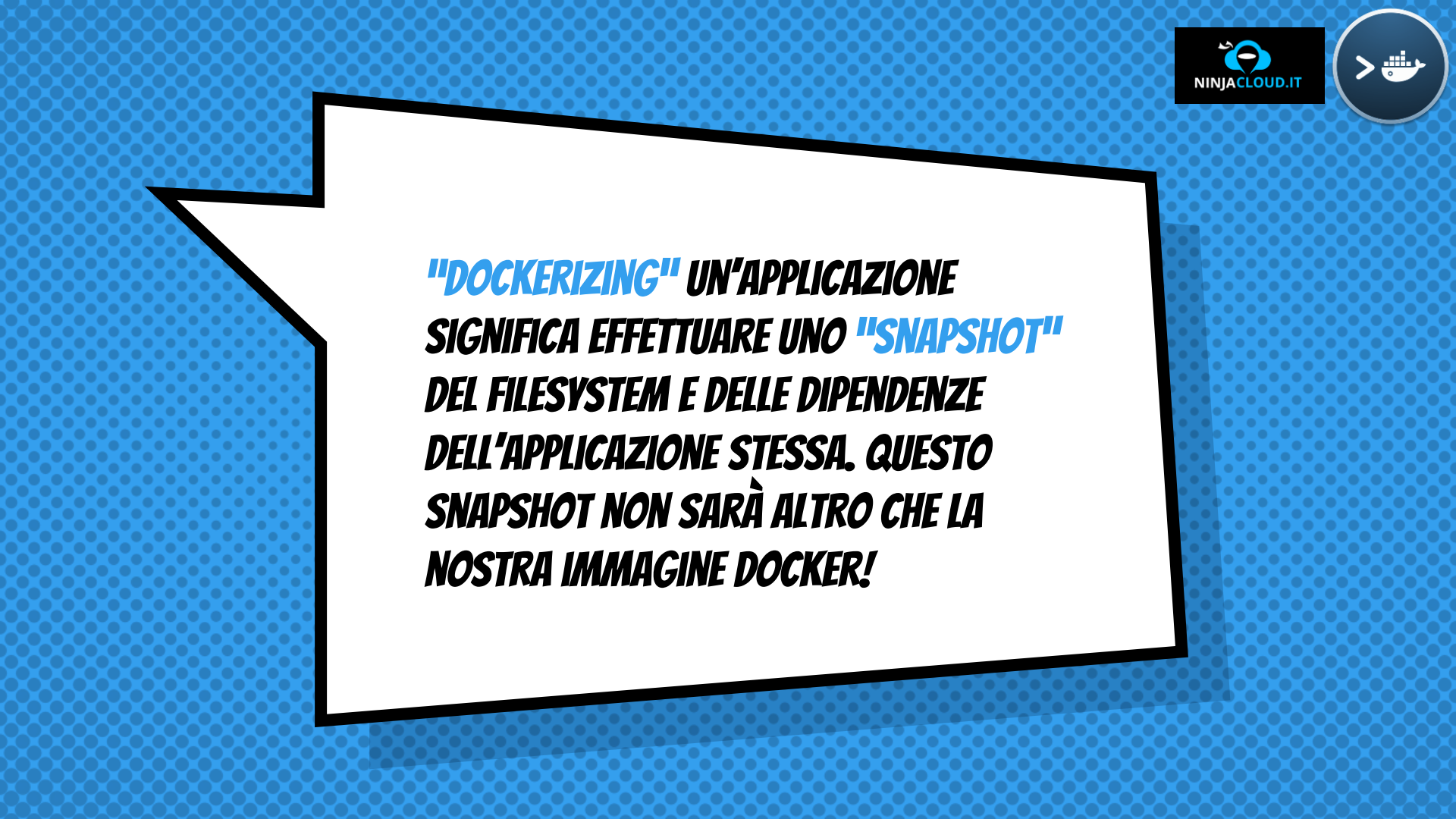


NINJACLOUD.IT

I CONTAINER ESEGUONO DELLE APPLICAZIONI O DEI SERVIZI QUINDI NASCE SPONTANEA L'ESIGENZA DI CONVERTIRE UN'APPLICAZIONE E RENDERLA UN CONTAINER.

NINJACLOUD.IT

QUESTO PROCESSO PRENDE IL NOME DI: "CONTAINERIZING" OPPURE "DOCKERIZING". IN ITALIANO NON SI RIESCE A TRADURRE CORRETTAMENTE.


NINJACLOUD.IT
"DOCKERIZING" UN'APPLICAZIONE SIGNIFICA EFFETTUARE UNO "SNAPSHOT" DEL FILESYSTEM E DELLE DIPENDENZE DELL'APPLICAZIONE STESSA. QUESTO SNAPSHOT NON SARÀ ALTRO CHE LA NOSTRA IMMAGINE DOCKER!


NINJACLOUD.IT
CREARE LE PROPRIE IMMAGINI GRAZIE ALLA DEFINIZIONE DI UN DOCKERFILE È UN CONCETTO FONDAMENTALE: CI PERMETTE DI PERSONALIZZARE LE NOSTRE IMMAGINI DOCKER SULLA BASE DELLA NOSTRA INFRASTRUTTURA ATTUALE.

PROCESSO DI "CONTAINERIZING" DI UN APP

Questo processo ad alto livello si può riassumere nelle seguenti fasi:

1. **Creazione dell'applicazione e accesso al codice della stessa.**
2. **Creazione del "dockerfile" che contiene tutte le informazioni dell'applicazione, le dipendenze e tutto ciò che sarà necessario per eseguirla.**
3. **Creazione dell'immagine derivata dal dockerfile.**
4. **Creazione ed esecuzione del container e quindi esecuzione della nostra applicazione.**

Tale processo si riassume nelle seguenti macro fasi:

BUILD -> SHIP -> RUN


1
CODICE APPLICAZIONE
2
CREAZIONE DOCKERFILE
3
CREAZIONE IMMAGINE
4
PUSHING IMMAGINE
5
IMMAGINE NEL REGISTRO
6
CREAZIONE CONTAINER

NINJACLOUD.IT

ARGOMENTO RICHIESTO AI FINI DELLA CERTIFICAZIONE!!

ESTRATTO DALLA STUDY GUIDE:

“Apply a file to create a Docker image”.

NINJACLOUD.IT

STRUTTURA DI UN DOCKERFILE

Come già introdotto, il dockerfile ci permette di definire la nostra immagine che, successivamente, tramite il comando **"docker image build"** sarà effettivamente creata ed utilizzata per avviare i container (docker container run).

Il dockerfile è un semplice file di testo che contiene tutte le istruzione affinché si possa creare l'immagine voluta.

Docker ci fornisce tutta una serie di comandi da utilizzare all'interno di questo file, tra cui: **FROM, CMD, EXPOSE, ENV, COPY, ADD**.

```
FROM golang:1.9.2-alpine3.6 AS build

# Install tools required for project
# Run `docker build --no-cache .` to update dependencies
RUN apk add --no-cache git
RUN go get github.com/golang/dep/cmd/dep

# List project dependencies with Gopkg.toml and Gopkg.lock
# These layers are only re-built when Gopkg files are updated
COPY Gopkg.lock Gopkg.toml /go/src/project/
WORKDIR /go/src/project/
# Install library dependencies
RUN dep ensure -vendor-only

# Copy the entire project and build it
# This layer is rebuilt when a file changes in the project director
COPY . /go/src/project/
RUN go build -o /bin/project

# This results in a single layer image
FROM scratch
COPY --from=build /bin/project /bin/project
ENTRYPOINT ["/bin/project"]
CMD ["--help"]
```

NINJACLOUD.IT

STRUTTURA DI UN DOCKERFILE

Quando si definisce una nuova immagine, è possibile partire completamente da zero oppure partire da un'immagine già presente, ciò è possibile farlo tramite l'istruzione FROM all'interno del dockerfile.

Prendiamo adesso come esempio l'immagine ALPINE e procediamo ad aggiungere dei tool non presenti nell'immagine base:

```
FROM alpine:latest

RUN apk update

RUN apk add vim
```

Il comando RUN esegue i comandi Linux esplicati. APK è il package manager della distribuzione ALPINE.

NINJACLOUD.IT

ARGOMENTO RICHIESTO AI FINI DELLA CERTIFICAZIONE!!

ESTRATTO DALLA STUDY GUIDE:

"Describe Dockerfile options [add, copy, volumes, expose, entrypoint, etc)".

NINJACLOUD.IT

ARGOMENTO RICHIESTO AI FINI DELLA CERTIFICAZIONE!!

ESTRATTO DALLA STUDY GUIDE:

“Show the main parts of a Dockerfile ”.

NINJACLOUD.IT

Partire con le definizione di un'immagine da zero è una pratica piuttosto complessa e avanzata. Strumenti come debootstrap, yumbootstrap e rinse si occupano proprio di questo.

FATTO CIÒ SI PROCEDE CON LA CREAZIONE EFFETTIVA DELL'IMMAGINE TRAMITE IL COMANDO "DOCKER IMAGE BUILD".

NINJACLOUD.IT

ANALIZZIAMO "DOCKER IMAGE BUILD"

Ci sono principalmente due modi di procedere alla creazione dell'immagine a partire dal dockerfile che ci siamo costruiti.

PRIMO MODO -> specificando il nome del file con l'opzione "--f" e dando un nome all'immagine con l'opzione "--t":

$ docker image build --file <path_to_Dockerfile> --tag <REPOSITORY>:<TAG> .

Solitamente l'opzione "--f" non si utilizza in quanto potrebbe creare dei problemi se abbiamo più file che devono essere inclusi. Una possibilità è quello di inserire il dockerfile all'interno di una cartella separata con ogni altro file che deve essere incluso all'interno dell'immagine:

$ docker image build --tag local:dockerfile-example .

Ricordasi il punto in fondo che specifica di considerare la cartella corrente per il "build"

NINJACLOUD.IT

ARGOMENTO RICHIESTO AI FINI DELLA CERTIFICAZIONE!!

ESTRATTO DALLA STUDY GUIDE:

"Inspect images and report specific attributes using filter and format ".

NINJACLOUD.IT

Notare che ogni cambiamento effettuato all'immagine base Alpine costituisce un **nuovo strato dell'immagine**. Ti invito a ripassare la sezione immagini e container se non ricordi a cosa mi sto riferendo.

NINJACLOUD.IT

NELLO SPECIFICO SOLO LE ISTRUZIONI: RUN, COPY E ADD CREANO DEI NUOVI STRATI (LAYERS) NELL'IMMAGINE.

NINJACLOUD.IT

Il fatto di creare questi strati è di grande vantaggio in quanto Docker può sfruttare un meccanismo di "cache" dell'immagine ed effettuare il build solo per i cambiamenti effettivi dell'immagine non ancora presenti nella sua cache interna.

NINJACLOUD.IT

PRINCIPALI COMANDI DI UN DOCKERFILE

FROM –> specifica l'immagine di partenza.

COPY/ADD –> copia file e cartelle dall'host verso l'immagine.

ENV –> permette la configurazione di variabili d'ambiente.

RUN –> esplica i comandi Linux da eseguire.

VOLUME –> definisce i volumi da utilizzare per la memorizzazione dei dati.

USER –> specifica un utente.

WORKDIR –> imposta la cartella di lavoro.

EXPOSE –> definisce le porte da esporre.

CMD –> specifica qual'è il processo principale che deve essere eseguito dall'immagine.

NINJACLOUD.IT

BEST PRACTICE RELATIVE AL DOCKERFILE

Di seguito alcuni suggerimenti utili relativi alla compilazione del docker file:

1. Utilizzare, quando possibile il **.dockerignore** file.
2. Usare il **"versioning"** così da organizzare meglio le varie versioni del dockerfile.
3. **Ridurre al minimo i pacchetti** necessari per ogni immagine. Tutto ciò che non è strettamente necessario non dev'essere installato.
4. Cercare di eseguire **una sola applicazione per container** così da mantenerlo semplice e ben gestibile.
5. **Fare riferimento alla documentazione** per ulteriori consigli su come strutturare il dockerfile.

NINJACLOUD.IT

ARGOMENTO RICHIESTO AI FINI DELLA CERTIFICAZIONE!!

ESTRATTO DALLA STUDY GUIDE:

"Give examples on how to create an efficient image via a Dockerfile".

NINJACLOUD.IT

Notare che è possibile creare un'immagine a partire da un container già configurato e convertito successivamente in un'altra immagine. Tuttavia questa pratica è sconsigliata ed è sempre preferibile utilizzare il Dockerfile.

PROCEDIAMO ADESSO ALLA DIMOSTRAZIONE PRATICA DI QUANTO DETTO.