



Pisa 21-02-2024

Containerization

Daniele Trezza

Senior java developer

Engineering studio

Research and development

Tutor



Nicholas Ferrari

Senior java developer

Cloud studio

Research and development

Menthor

- 33 Countries
- 5 Continents
- +29K Globers



We are behind the **digital transformations of brands like**

Google Santander EA Disney

The image shows a modern office lobby. On the left, there's a grey wall with the 'Globant' logo in black text and a yellow arrow pointing to the right. To the right of the wall is a green modular sofa. In the background, there's a white map of Europe with several colored location pins: United Kingdom (blue), Germany (orange), Poland (red), Lithuania (green), Romania (yellow), France (purple), Spain (pink), and Italy (light blue). The lobby also features a large orange sofa, some potted plants, and a window looking out onto a cityscape.

Be kind culture

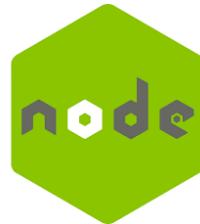
Career opportunities in +3500 projects worldwide

MA I CONTAINER... SERVONO?

Era meglio prima (?)

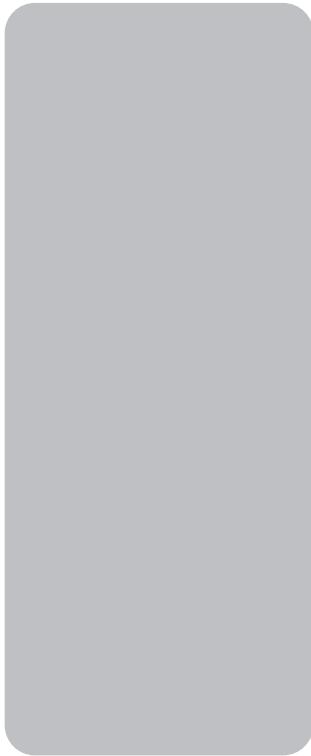
MA I CONTAINERS... SERVONO?

VMs VS Containers



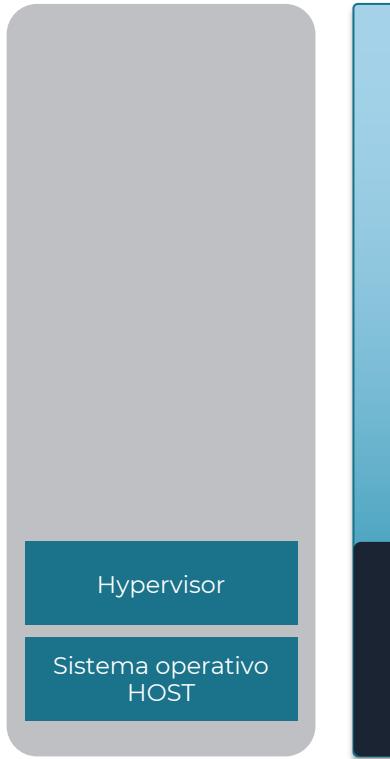
MA I CONTAINERS... SERVONO?

VMs VS Containers



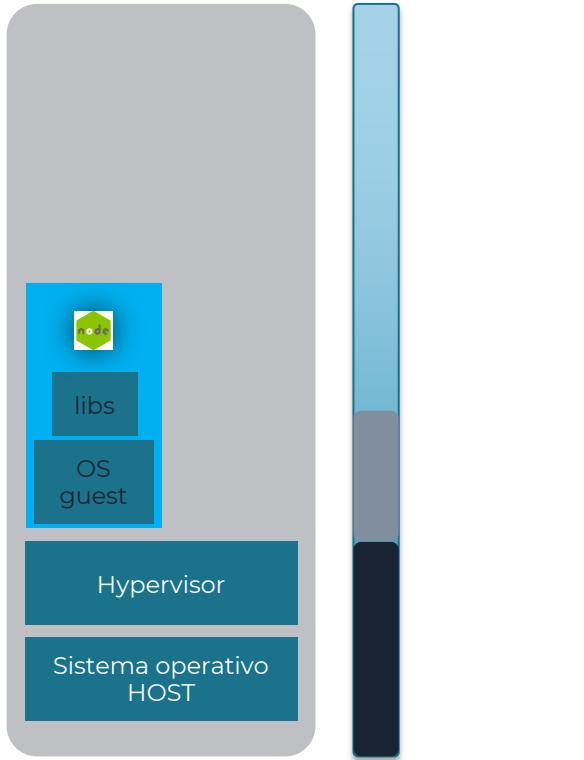
MA I CONTAINERS... SERVONO?

VMs VS Containers



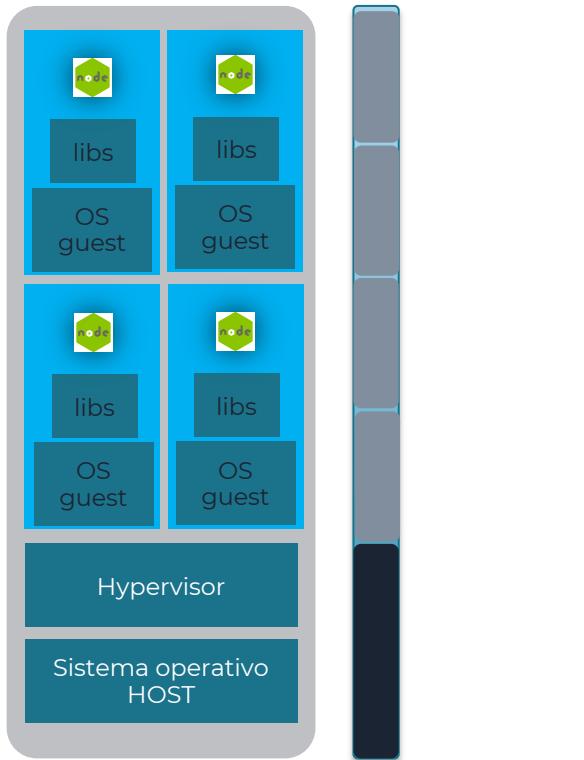
MA I CONTAINERS... SERVONO?

VMs VS Containers



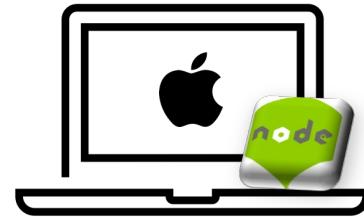
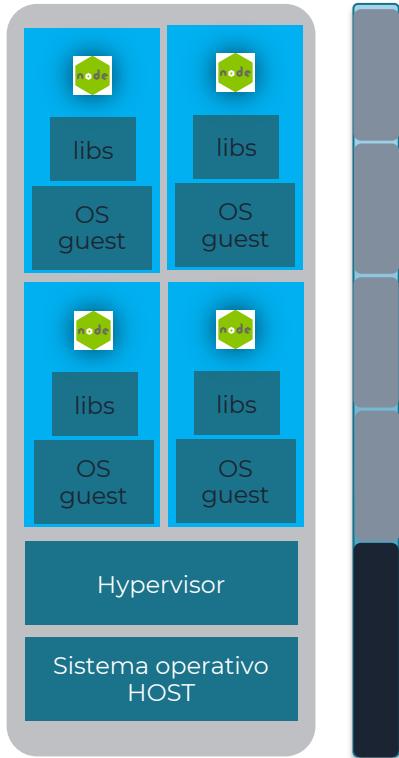
MA I CONTAINERS... SERVONO?

VMs VS Containers

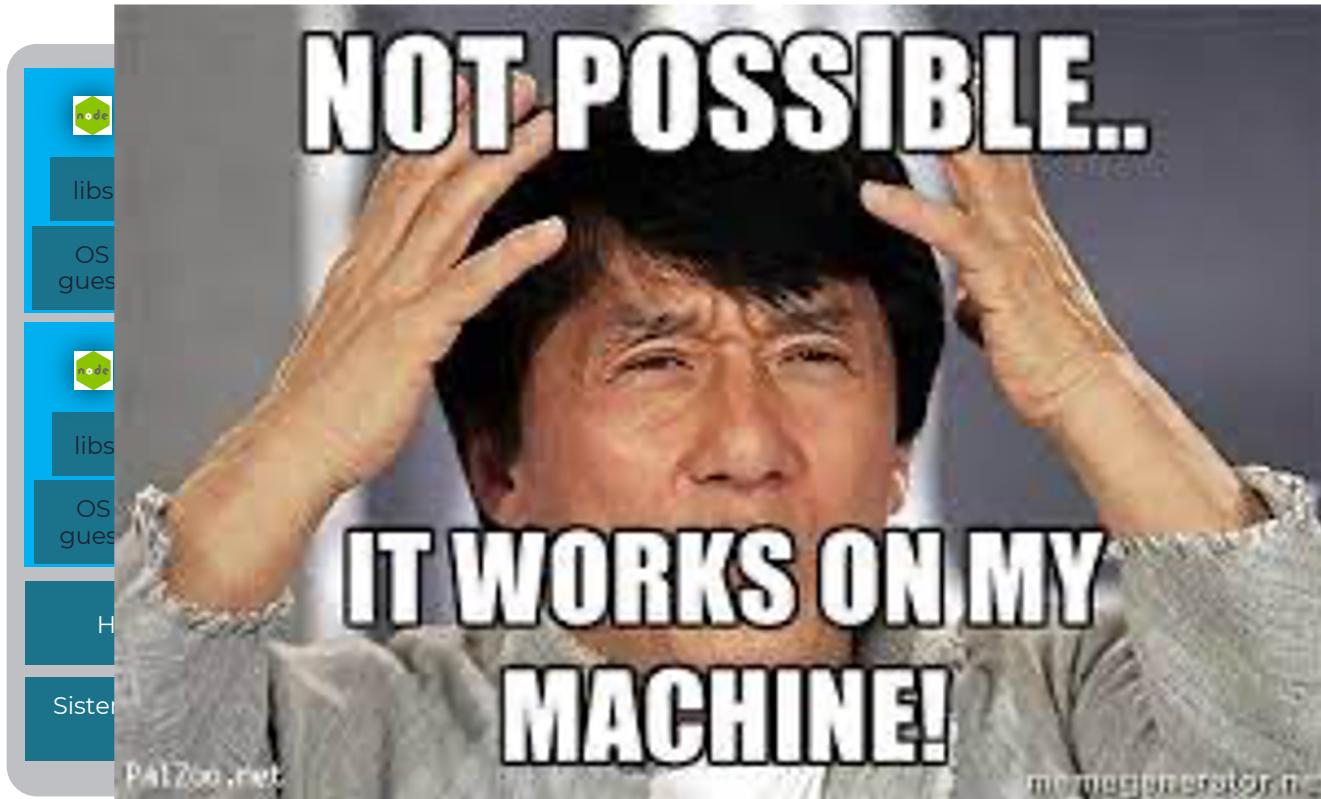


MA I CONTAINERS... SERVONO?

VMs VS Containers

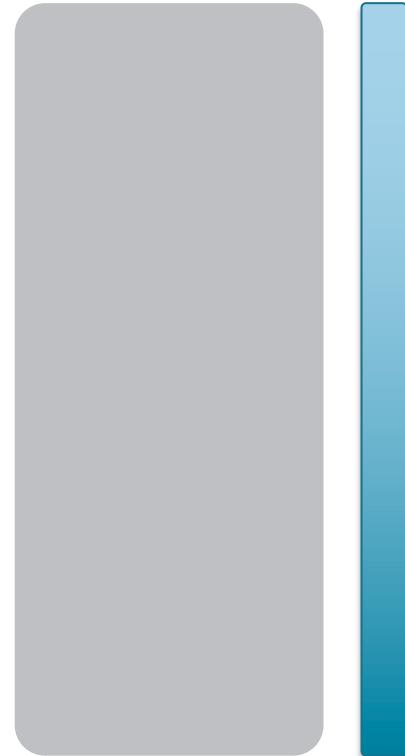
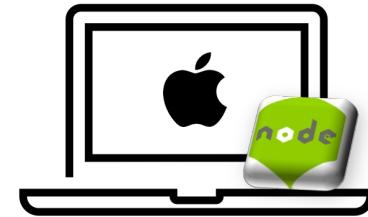
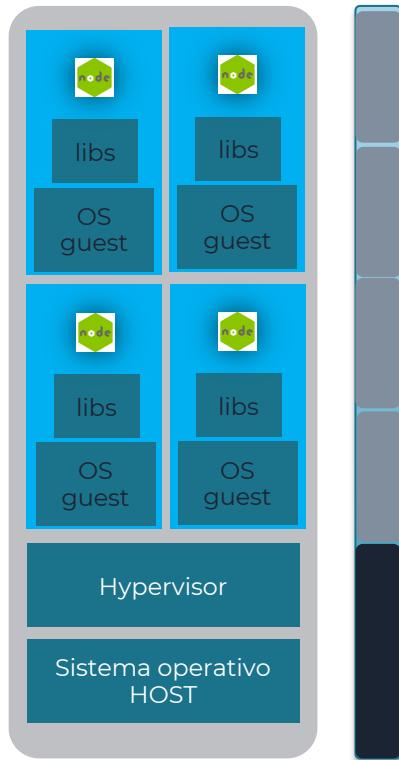


VMs VS Containers



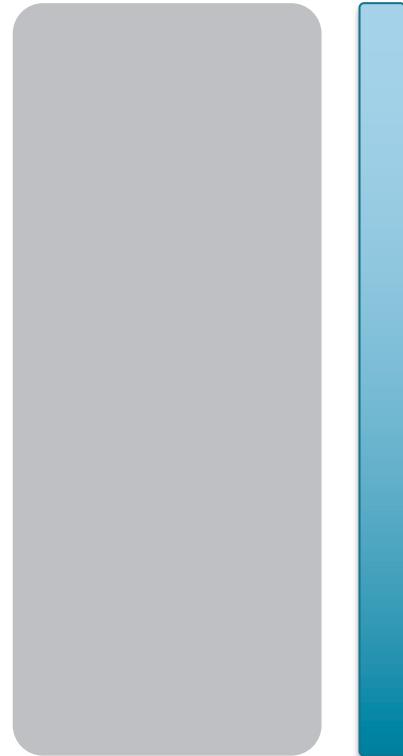
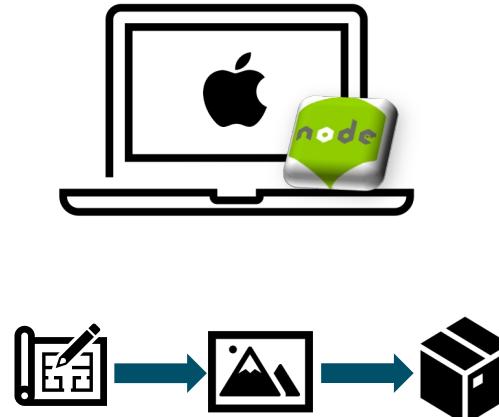
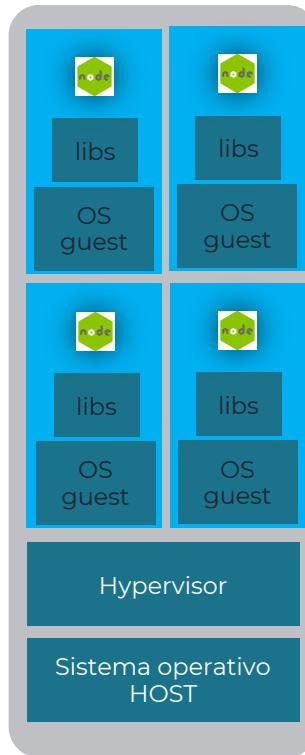
MA I CONTAINERS... SERVONO?

VMs VS Containers



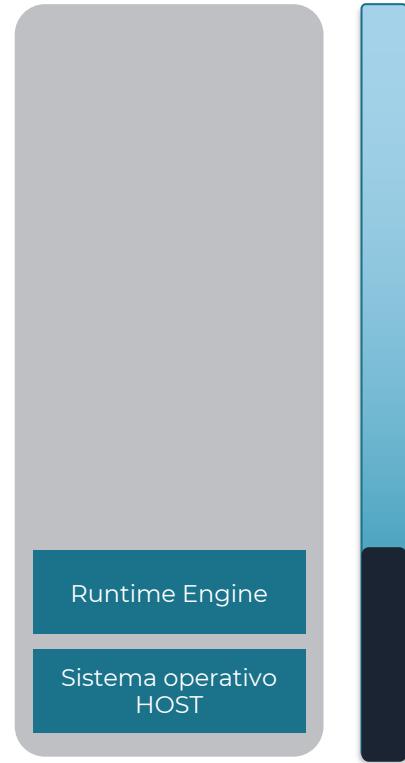
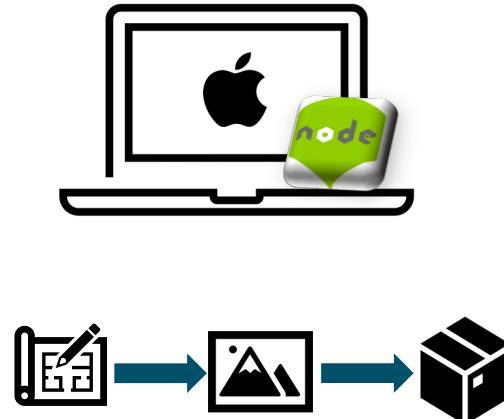
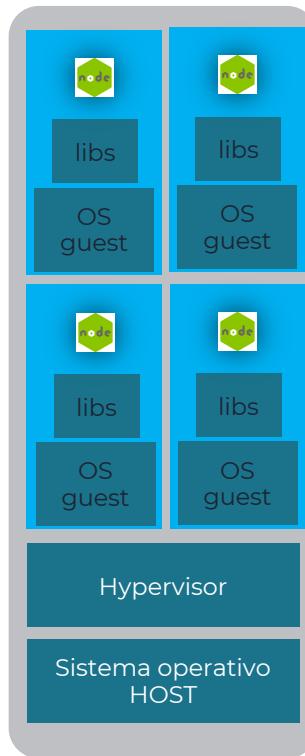
MA I CONTAINERS... SERVONO?

VMs VS Containers



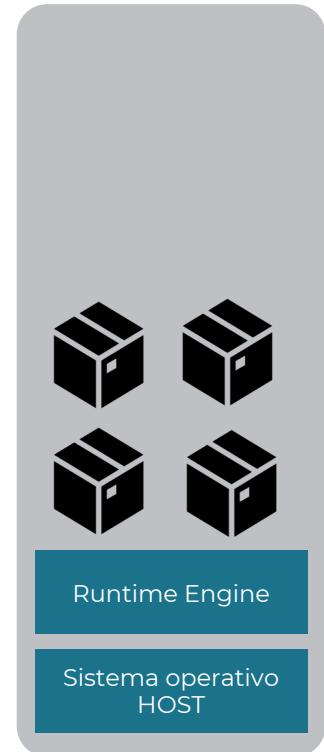
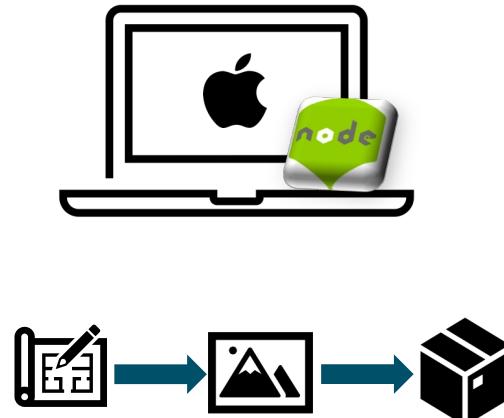
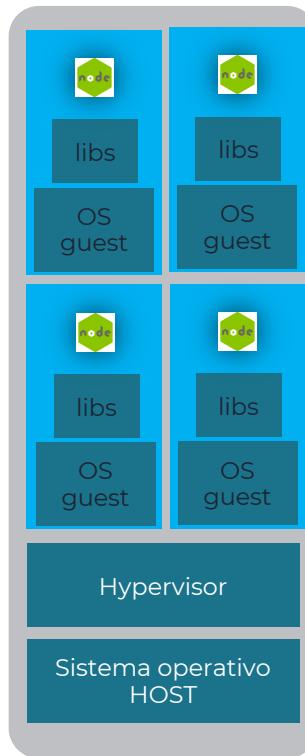
MA I CONTAINERS... SERVONO?

VMs VS Containers



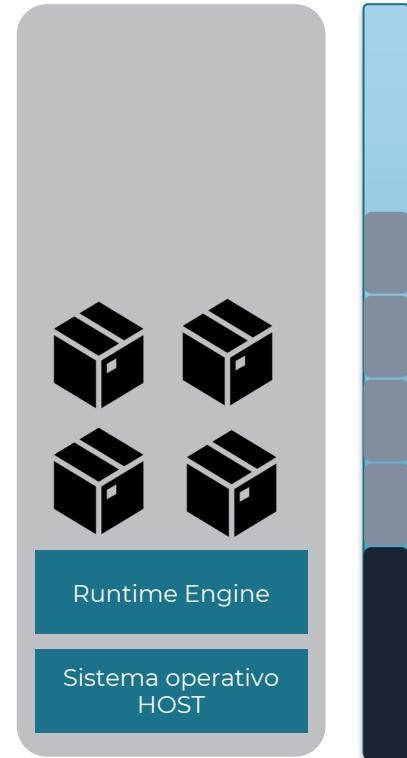
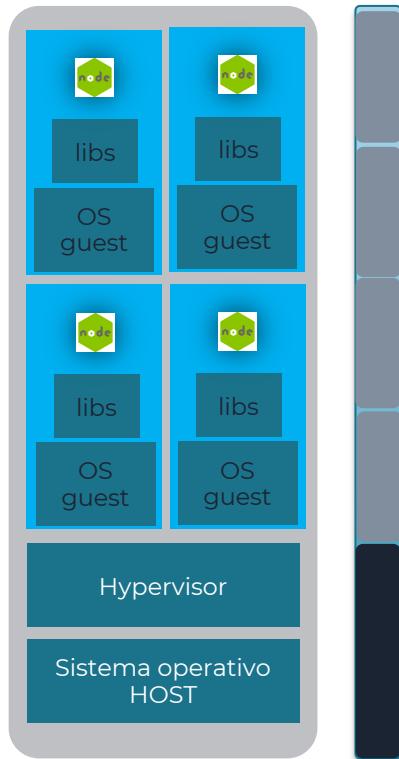
MA I CONTAINERS... SERVONO?

VMs VS Containers



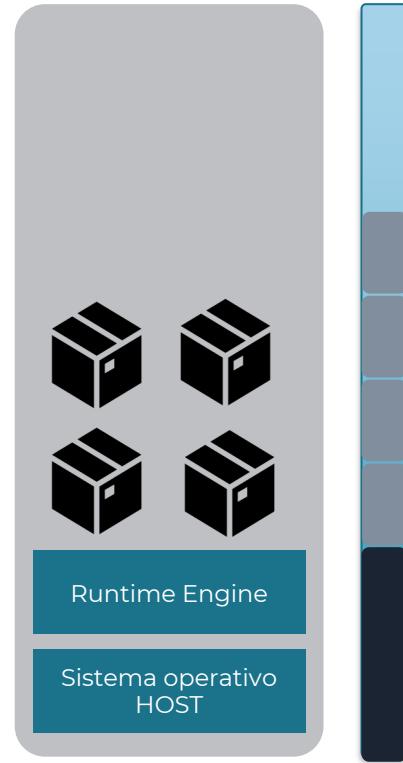
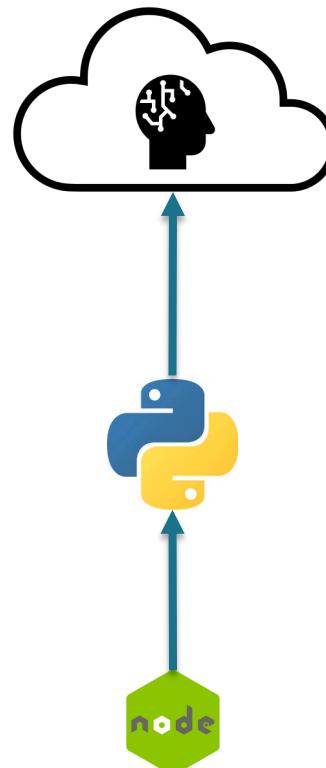
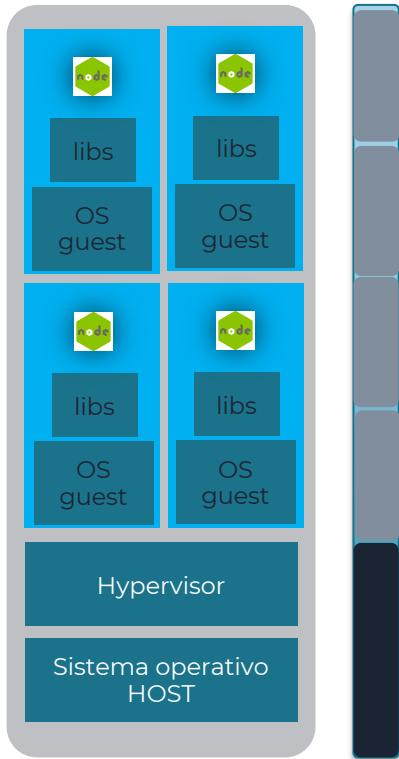
MA I CONTAINERS... SERVONO?

VMs VS Containers



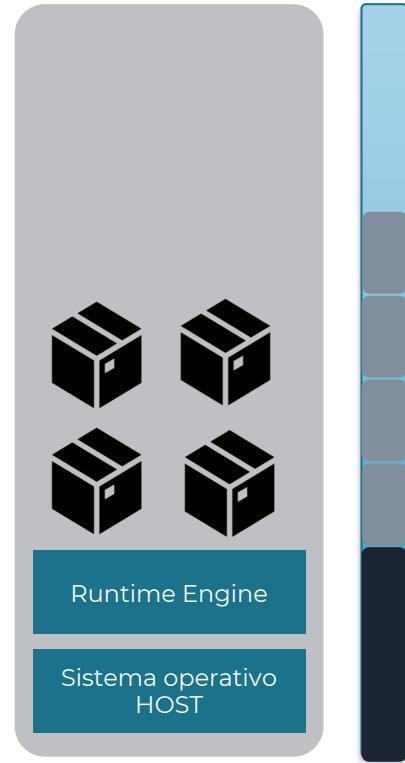
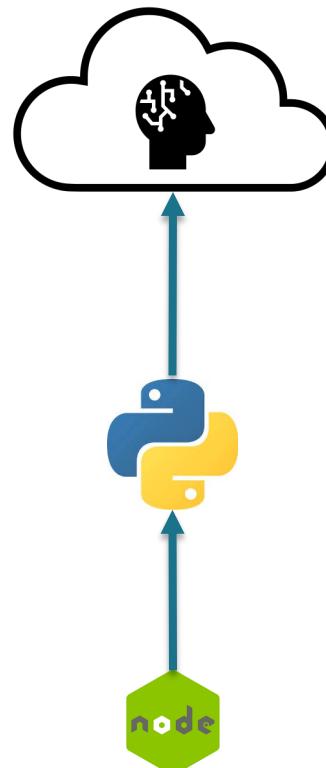
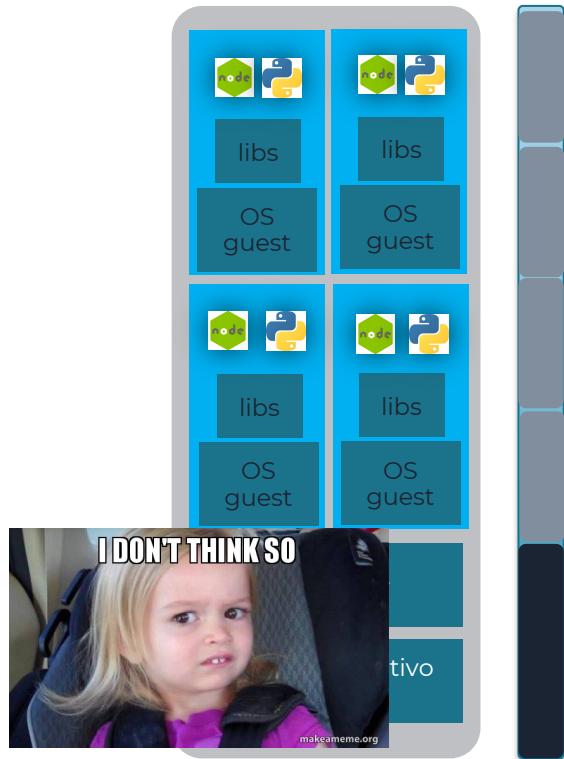
MA I CONTAINERS... SERVONO?

VMs VS Containers



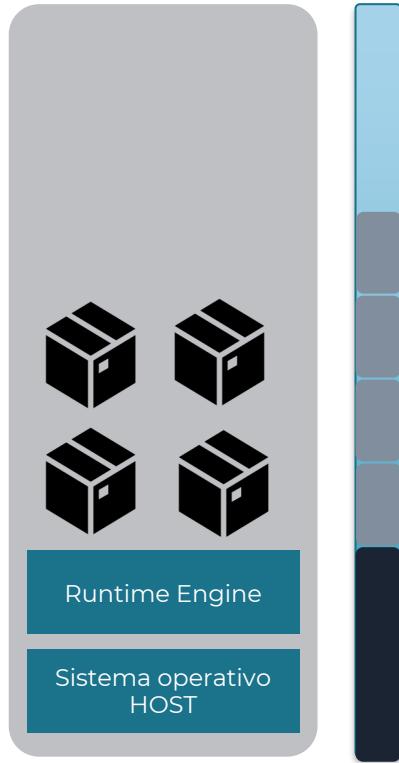
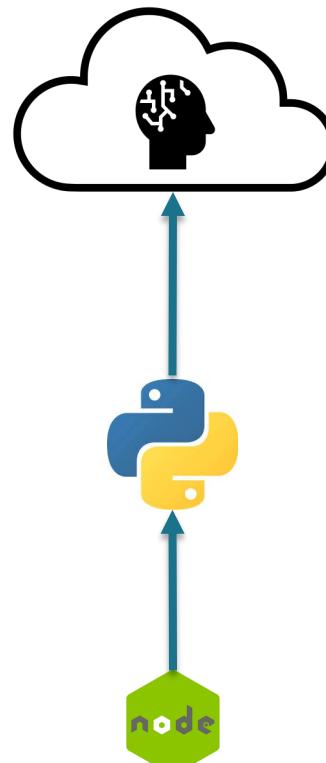
MA I CONTAINERS... SERVONO?

VMs VS Containers



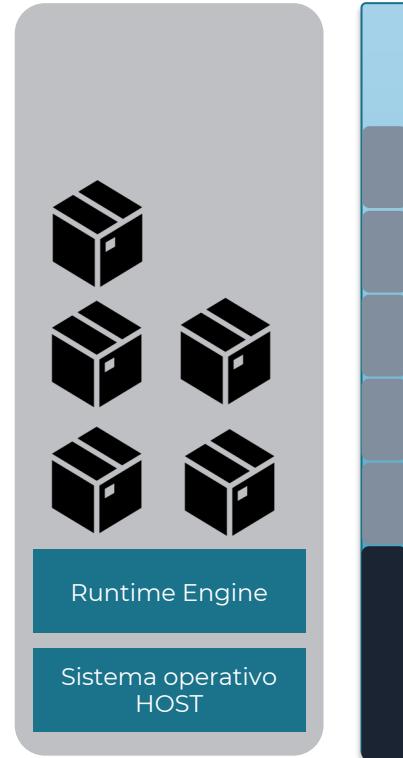
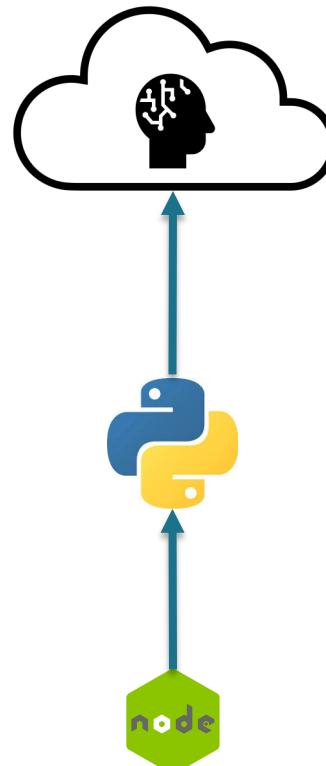
MA I CONTAINERS... SERVONO?

VMs VS Containers

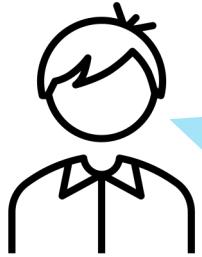


MA I CONTAINERS... SERVONO?

VMs VS Containers



Costi



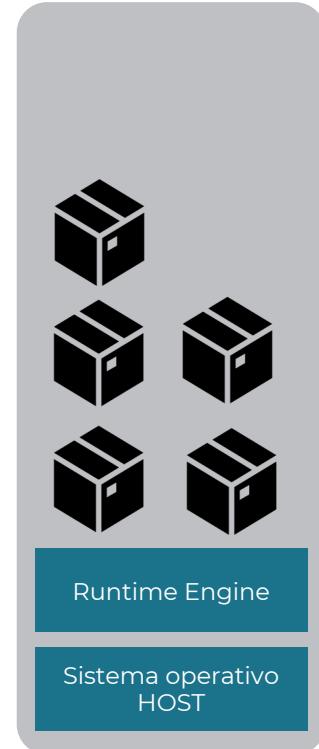
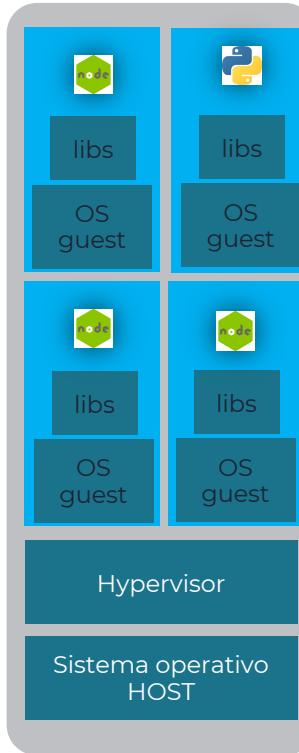
Ma io sto sul cloud...
Non ho hardware...



ma che ce ne fooott

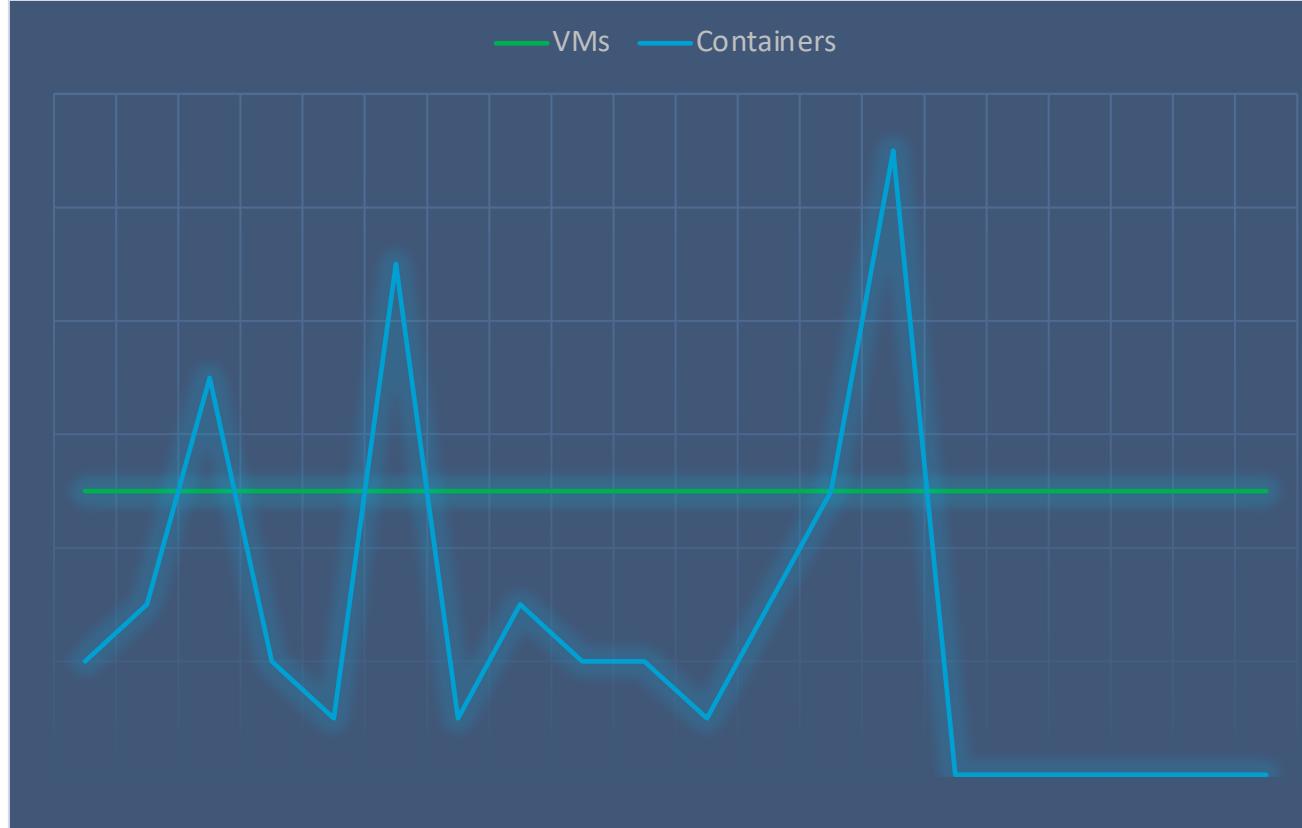
MA I CONTAINER... SERVONO?

Costi

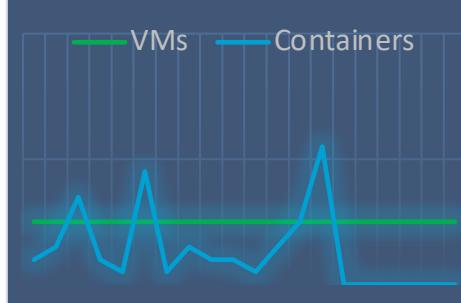


MA I CONTAINER... SERVONO?

Costi



Costi



Per il programmatore:

- Pensa modulare
- Test semplificato
- Troubleshooting rapido
- Divisione delle responsabilità

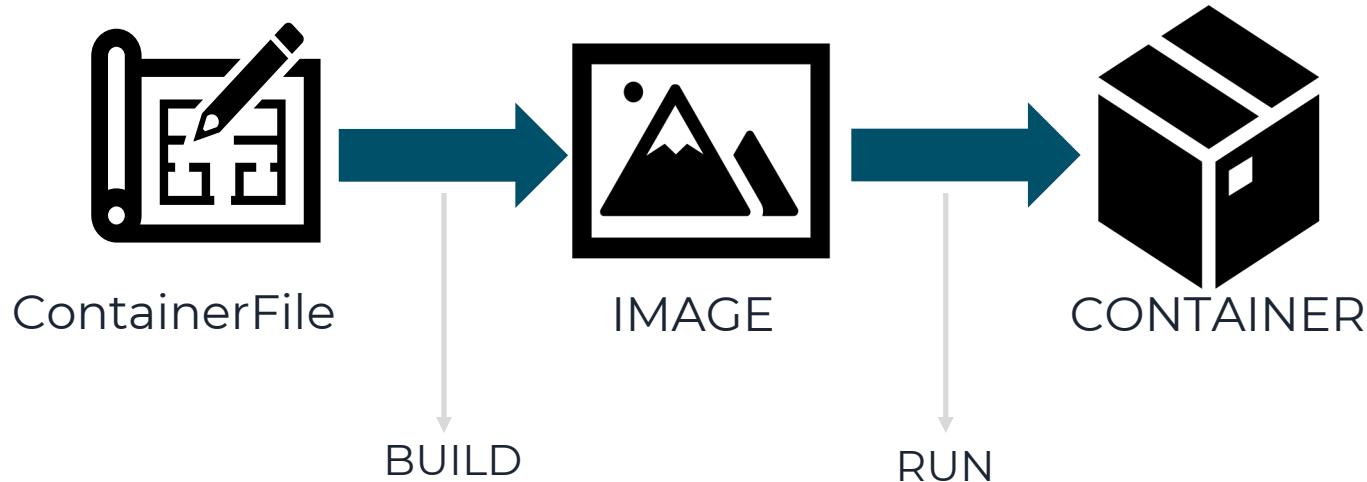
TERMINOLOGIA



Il vocabolario base



From the image to the container



TERMINOLOGIA

Layering



TERMINOLOGIA

Layering



Layering



Layering



```
[REDACTED]:~$ docker pull postgres
Using default tag: latest
latest: Pulling from library/postgres
8d691f585fa8: Pull complete
c991029393ff: Pull complete
d104c69c9175: Pull complete
0a7fb105514d: Pull complete
c3d11c21cb77: Pull complete
4536342c5414: Pull complete
435bcefd4e05: Pull complete
36b0869ae6f9: Pull complete
5ac554d17b78: Pull complete
61foa5a69de4: Pull complete
f3613132ea9e: Pull complete
8d022c339281: Pull complete
29616bd9cc5c: Pull complete
6283090fa09d: Pull complete
Digest: sha256:a4a944788084a92bcaff6180833428f17cce610e43c828b3a42345b33a608a7
Status: Downloaded newer image for postgres:latest
[REDACTED]:~$
```

```
[REDACTED]:~$ docker pull mariadb
Using default tag: latest
latest: Pulling from library/mariadb
22e816666fd6: Already exists
079bd6d2a1e53: Already exists
11048eba9098: Already exists
c58094023a2e: Already exists
le8f13102fa0: Pull complete
8c1425d731a6: Pull complete
14e6f69e6aab: Pull complete
c90c2f3858cf: Pull complete
b78202ba9229: Pull complete
cadce28d1b9c: Pull complete
6fb2c5af5492: Pull complete
7a59522b36b8: Pull complete
722b05c4c4b1: Pull complete
bd4039b5406f: Pull complete
Digest: sha256:7b371982ac83beee40bee645c6efab1bc9113abbce9cbc95bf3eddac268adf57
Status: Downloaded newer image for mariadb:latest
[REDACTED]:~$
```

Layering



TERMINOLOGIA

Layering



TERMINOLOGIA

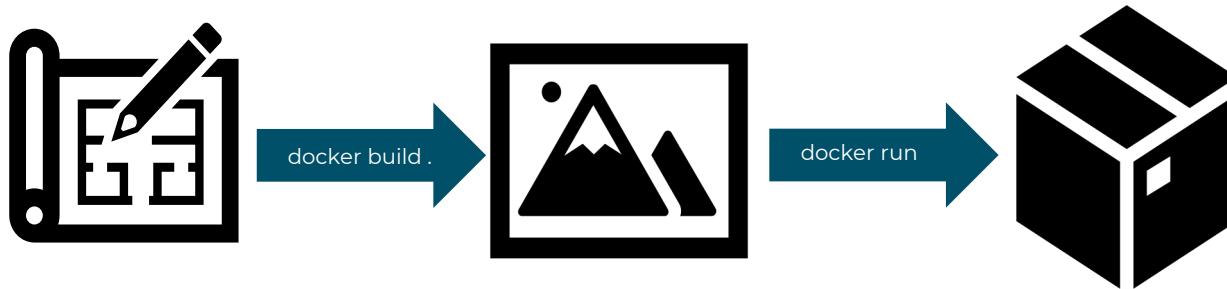
Layering



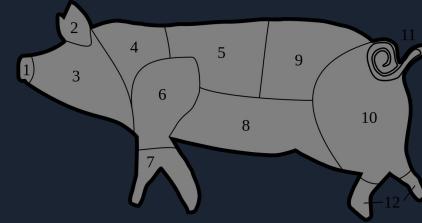
From the image to the container



From the image to the container



ANATOMY OF A CONTAINERFILE



Basic commands

```
FROM node:13-alpine
```

```
ENV MONGO_DB_USERNAME=admin \  
    MONGO_DB_PWD=password
```

```
RUN mkdir -p /home/app
```

```
COPY . /home/app
```

```
CMD ["node", "/home/app/server.js"]
```

Basic commands

```
FROM node:13-alpine
```

Ogni containerfile ha sempre come primo comando il FROM e il nome di una **immagine da cui partire**.

```
ENV MONGO_DB_USERNAME=admin \  
MONGO_DB_PWD=password
```

La definizione di variabili di ambiente è **opzionale**.
Queste variabili saranno visibili solo **dall'interno** del container.

```
RUN mkdir -p /home/app
```

Permette di eseguire **qualsiasi comando** linux, **all'interno** del container.

```
COPY . /home/app
```

Copia i file **dall'host al container**.
Unico comando che agisce anche sull'HOST.

```
CMD ["node", "/home/app/server.js"]
```

Il **primo comando** da eseguire quando parte il container.

PRO commands

ARG : il FROM può prendere degli argomenti dichiarati dalle istruzioni ARG che lo precedono.

ARG può essere usato anche per dichiarare delle variabili interne.

LABEL : aggiunge metadati.

EXPOSE : informa che il container ascolterà su una determinata porta. NON pubblica la porta. Per pubblicare la porta si usa l'opzione –p quando si fa la run.

ADD : è un copy più potente. Si può specificare un URL da copiare nel filesystem del container. Se si passa come primo argomento un file compresso, questo verrà decompresso nel container.

ENTRYPOINT : comando da eseguire quando il container parte. Senza Entrypoint di default si userà /bin/sh -c bash. CMD rimane obbligatorio, e rappresenta il parametro di default.

VOLUME : specifica il volume, ossia dove va mantenuto lo stato del container.

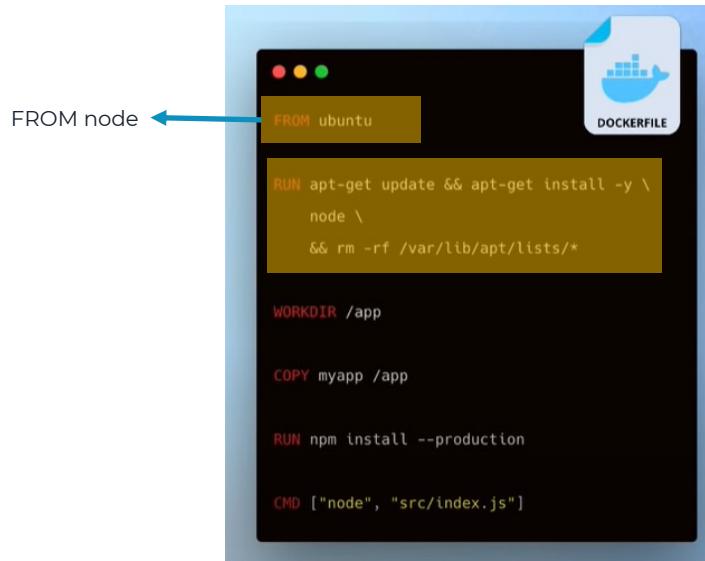
WORKDIR : setta la directory di lavoro per le istruzioni che lo seguono.

ONBUILD : Utile per immagini che so saranno usate per derivare altre immagini. Permette di impostare i primi comandi che verranno eseguiti alla creazione della prossima immagine. Vedi Maven.

BEST PRACTICES



1 - USA IMMAGINI DI PARTENZA UFFICIALI E VICINE ALLA TUA NECESSITA'



Così il containerFile viene più pulito, e in più l'immagine di partenza sarà già costruita seguendo le best practices

2 - EVITA IL LATEST

Non puoi prevedere che la versione latest supporterà sempre le tue funzionalità

Due build diverse fatte in momenti diversi potrebbero generare risultati diversi



A screenshot of a Dockerfile editor window. The Dockerfile contains the following code:

```
FROM ubuntu
RUN apt-get update && apt-get install -y \
    node \
&& rm -rf /var/lib/apt/lists/*
WORKDIR /app
COPY myapp /app
RUN npm install --production
CMD ["node", "src/index.js"]
```

The line `FROM node:latest` is highlighted with a green background and a yellow border. A blue arrow points from the text "FROM node:latest" in the first paragraph to this highlighted line in the Dockerfile.

3 - OTTIMIZZA LA TUA CACHE

Ho un containerFile in cui prima faccio la copia della mia app node e poi installo le dipendenze.



A screenshot of a terminal window displaying a Dockerfile. The file content is as follows:

```
FROM node:17.0.1-alpine

WORKDIR /app

COPY myapp /app

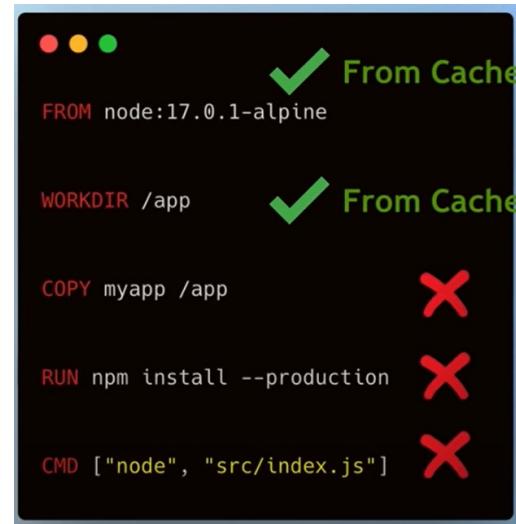
RUN npm install --production

CMD ["node", "src/index.js"]
```

The line `COPY myapp /app` is highlighted with a green rectangular background.

3 - OTTIMIZZA LA TUA CACHE

Se faccio una modifica al codice, MA non alle dipendenze, il comando npm install non verrà preso da cache!



I layer che stanno sopra al layer modificato vengono tutti cancellati dalla cache

3 - OTTIMIZZA LA TUA CACHE

Vorrei che npm install venga eseguito solo quando il file package.json viene modificato

Prima copio il file package, faccio npm install e alla fine copio dentro l'applicazione.

```
FROM node:17.0.1-alpine

WORKDIR /app

COPY package.json package-lock.json .

RUN npm install --production

COPY myapp /app

CMD ["node", "src/index.js"]
```

IN GENERALE: Ordinare i comandi dal meno soggetto a cambiamenti al più soggetto a cambiamenti

4 - IL FILE `.containerignore` è tuo amico `.dockerignore`

Una applicazione in run non ha bisogno di tutti i file che abbiamo sul repository....

```
# ignore .git and .cache folders
.git
.cache

# ignore all markdown files (md)
*.md

# ignore sensitive files
private.key
settings.json
```

CARTELLE AUTO-GENERATE... README... CARTELLA DI GIT...

5 – MULTI-STAGE BUILDS

Ci sono contenuti di cui hai bisogno in fase di build, ma non quando l'applicazione gira
DEVELOPMENT TOOLS...BUILD TOOLS...TEST DEPENDENCIES... TEMPORARY FILES

Esempi:

In una Java Application, hai bisogno della **JDK** solo per compilarla, ma non per farla girare.
Maven o **Gradle** non sono necessari quando in run.

```
FROM tomcat
RUN apt-get update \
    && apt-get -y install maven
WORKDIR /app
COPY myapp /app
RUN mvn package

COPY --from=build /app/target/file.war /usr/local/tomcat/webapps
EXPOSE 8080
ENTRYPOINT ["java","-jar","/usr/local/lib/demo.jar"]
```

How do we exclude dependencies from

COME SI FA A SEPARARE LA
FASE DI BUILD DALLA FASE DI
RUNTIME?

5 – MULTI-STAGE BUILDS

Ci sono contenuti di cui hai bisogno in fase di build, ma non quando l'applicazione gira
DEVELOPMENT TOOLS...BUILD TOOLS...TEST DEPENDENCIES... TEMPORARY FILES

The diagram illustrates a multi-stage Dockerfile structure. It features two vertical columns. The left column, labeled '1st' at the top, represents the build stage. The right column, labeled '2nd' at the top, represents the run stage. The build stage contains commands for building a Maven application: FROM maven AS build, WORKDIR /app, COPY myapp /app, and RUN mvn package. The run stage starts with FROM tomcat, followed by COPY --from=build /app/target/file.war /usr/local/tomcat/webapps/. The ellipsis '...' indicates additional steps.

```
# Build stage
FROM maven AS build
WORKDIR /app
COPY myapp /app
RUN mvn package

#Run stage
FROM tomcat
COPY --from=build /app/target/file.war /usr/local/tomcat/webapps/
...
```

Ogni From inizia una immagine nuova, con –from= build richiamo l'artifact di una immagine precedente.

Solo l'ultima sarà il risultato della build

6 – USA UTENTI CON POCHI PRIVILEGI

Soprattutto se stai usando *docker*...

CON UN UTENTE CREATO AD-HOC

```
...  
  
# create group and user  
RUN groupadd -r tom && useradd -g tom tom  
  
# set ownership and permissions  
RUN chown -R tom:tom /app  
  
# switch to user  
USER tom  
  
CMD node index.js
```

ALCUNE IMMAGINI INCLUDONO GIÀ UN UTENTE

```
FROM node:10-alpine  
  
...  
  
# set ownership and permissions  
RUN chown -R node:node /app  
  
# switch to node user  
USER node  
  
CMD ["node", "index.js"]
```

7 – SCAN con SNYC



```
x Low severity vulnerability found in freetype/freetype
Description: CVE-2020-15999
Info: https://snyk.io/vuln/SNYK-ALPINE310-FREETYPE-1019641
Introduced through: freetype/freetype@2.10.0-r0, gd/libgd@2.2.5-r2
From: freetype/freetype@2.10.0-r0
From: gd/libgd@2.2.5-r2 > freetype/freetype@2.10.0-r0
Fixed in: 2.10.0-r1

x Medium severity vulnerability found in libxml2/libxml2
Description: Out-of-bounds Read
Info: https://snyk.io/vuln/SNYK-ALPINE310-LIBXML2-674791
Introduced through: libxml2/libxml2@2.9.9-r3, libxslt/libxslt@1.1.33-r3, nginx-module-xslt/nginx-module-xslt@1.17.9-r1
From: libxml2/libxml2@2.9.9-r3
From: libxslt/libxslt@1.1.33-r3 > libxml2/libxml2@2.9.9-r3
From: nginx-module-xslt/nginx-module-xslt@1.17.9-r1 > libxml2/libxml2@2.9.9-r3
Fixed in: 2.9.9-r4
```



Si può configurare in molti repository, o integrarlo nella pipeline ci CI/CD

PODMAN



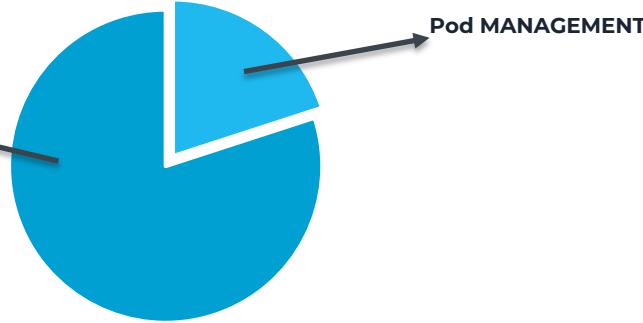
BASIC COMMANDS



```
alias docker=podman
```

Docker-inspired

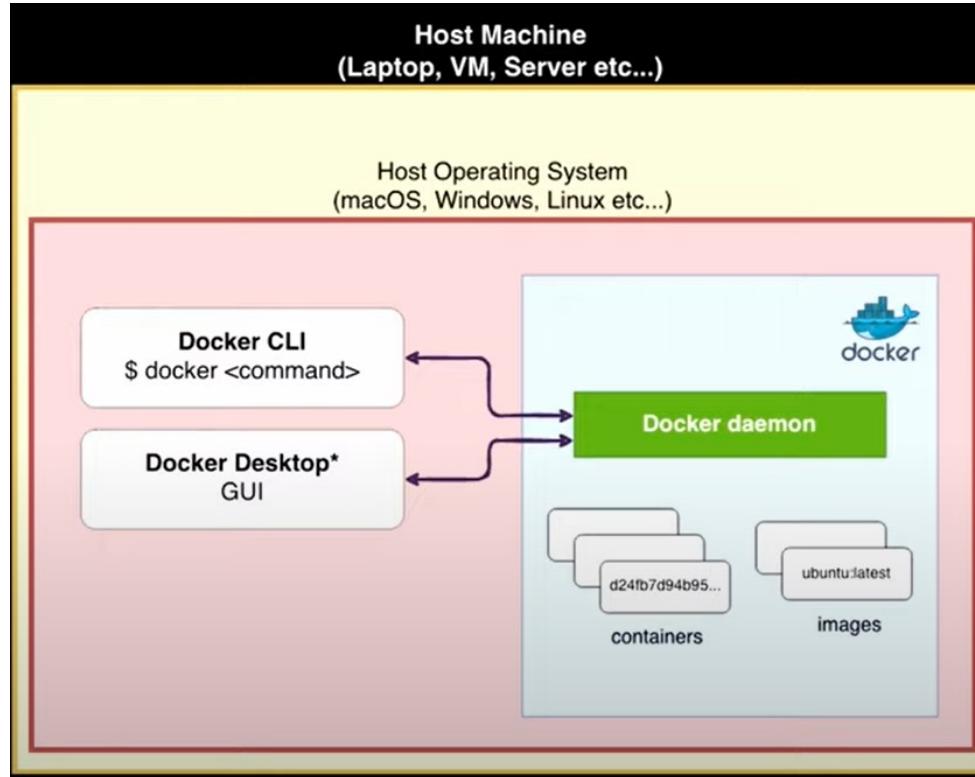
- Image build, update destroy
- Container run, stop
- Upload images to container registry



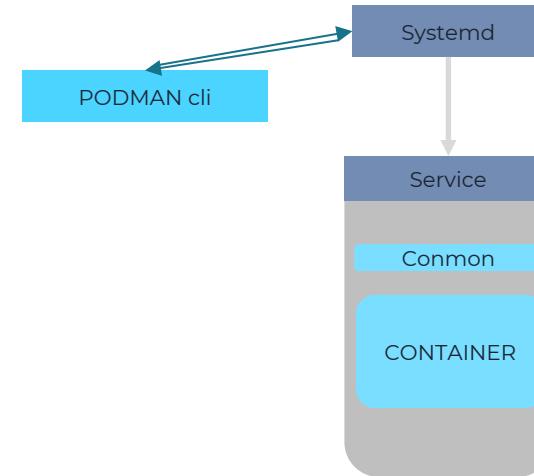
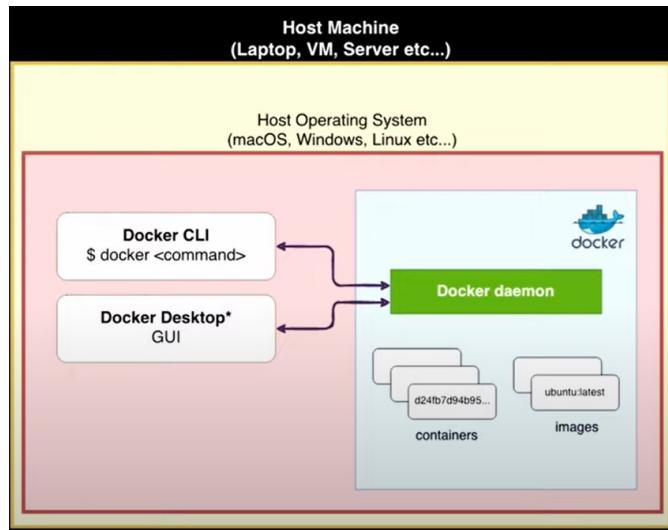
Ma allora... cosa c'è di diverso?

**DAEMON
ROOT LESS**

Ma allora... cosa c'è di diverso?



Ma allora... cosa c'è di diverso?

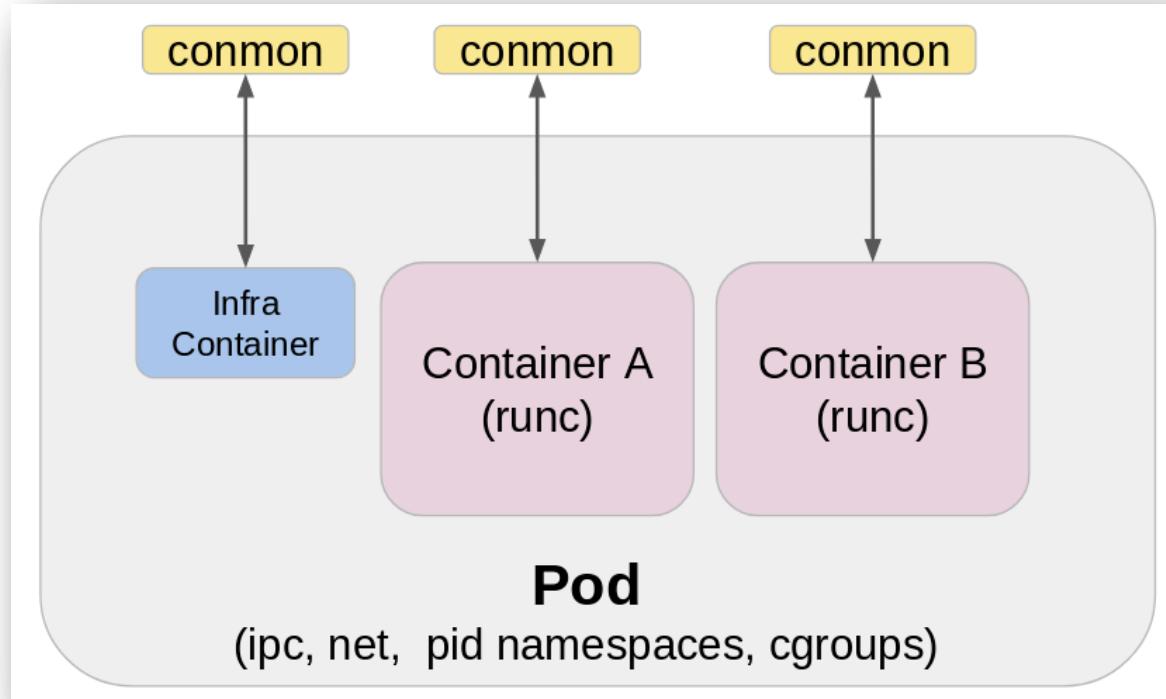


Docker si paga!!!

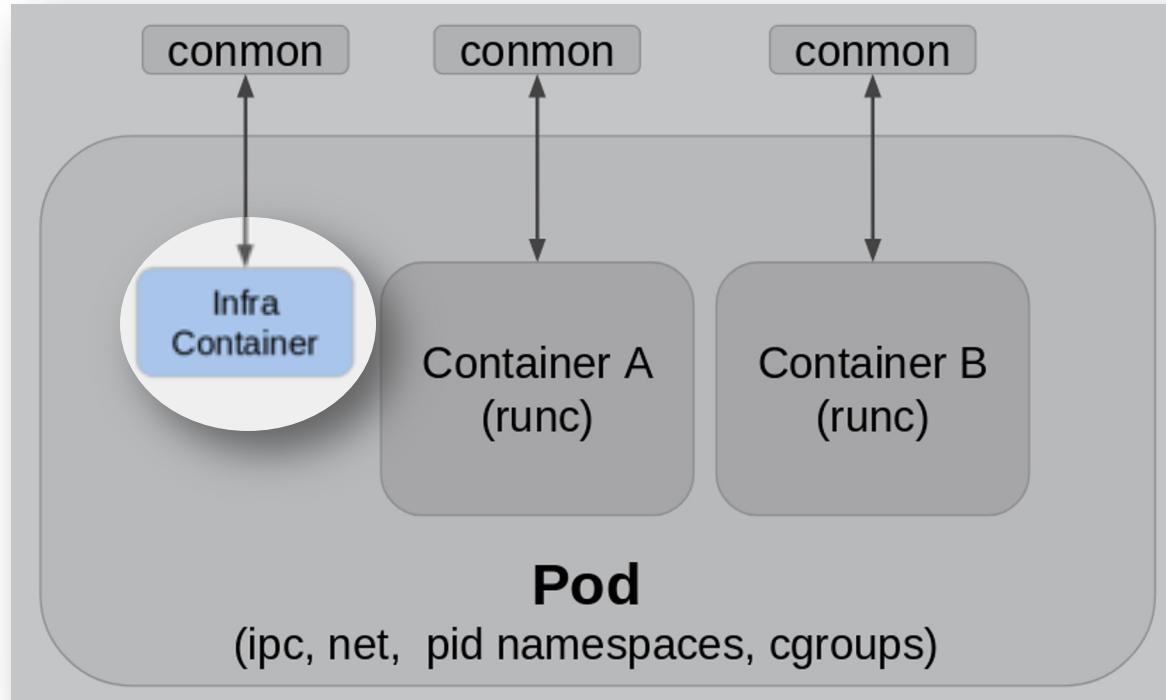
The screenshot shows the Docker pricing page with four subscription plans:

- Personal** (\$0): Ideal for individual developers, education, open source communities, and small businesses. Includes Docker Desktop, unlimited public repositories, Docker Engine + Kubernetes, and limited image pulls per day.
- Pro** (\$5/month): Includes pro tools for individual developers who want to accelerate their productivity. Everything in Personal plus:
 - Docker Desktop
 - Unlimited private repositories
 - 5,000 image pulls per day
 - 5 concurrent builds
 - 300 Hub vulnerability scans
 - 5 scoped access tokens
- Team** (\$7 /user/month): Ideal for teams and includes capabilities for collaboration, productivity and security. DEVELOPER FAVORITE Everything in Pro, plus:
 - Docker Desktop
 - Unlimited teams
 - 15 concurrent builds
 - Unlimited image scans
 - Unlimited scoped tokens
 - Role-based access control
 - Audit logsBilled annually starting at \$300.
- Business** (\$21 /user/month): Ideal for medium and large businesses who need centralized management and advanced security capabilities. Everything in Team, plus:
 - Docker Desktop
 - Centralized management
 - Image Access Management
 - SAML SSO *coming soon
 - Purchase via invoiceOnly available with an annual subscription.

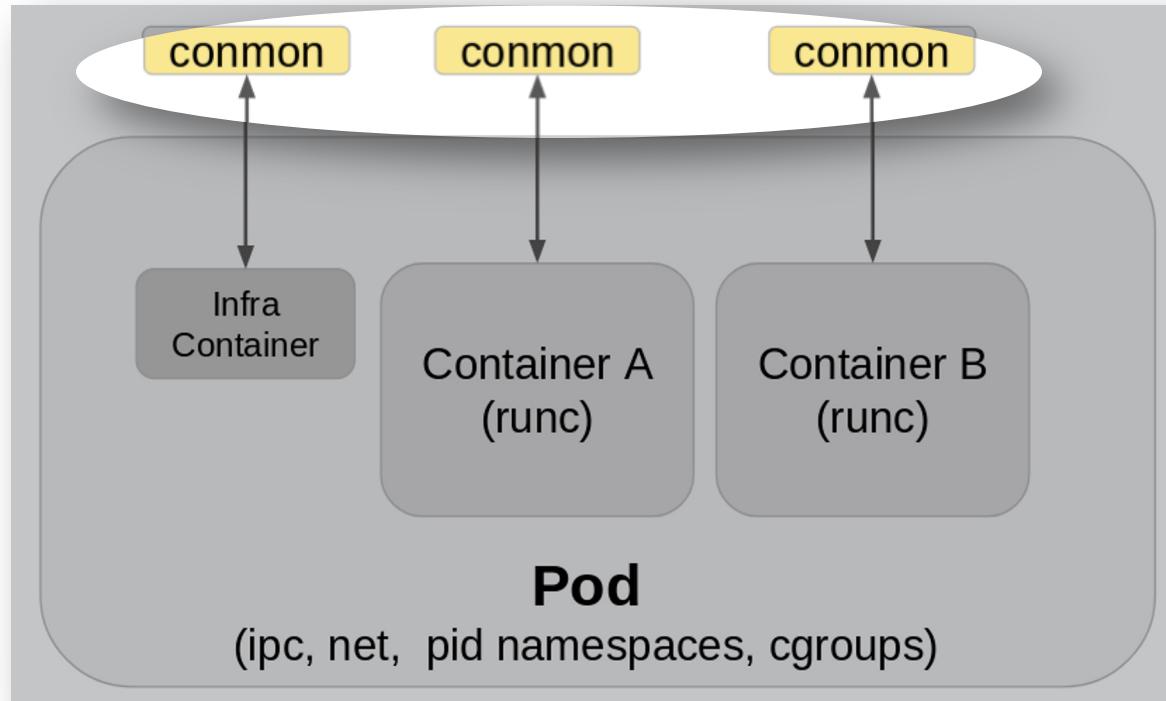
I POD



I POD



I POD



I POD

```
dtrezza@SYS-DTREZZA:/mnt/c/Users/Daniele$ podman pod create --name pod_di_daniele  
d3b4d7e6272dd720ee511d1efecc0a78b2203fbed06203579ab5d5069dfc4f5b
```

```
dtrezza@SYS-DTREZZA:/mnt/c/Users/Daniele$ podman pod ps  
POD ID          NAME           STATUS      CREATED          INFRA ID      # OF CONTAINERS  
d3b4d7e6272d  pod_di_daniele  Created    26 seconds ago  1b9a8d346182  1
```

```
podman run --pod pod_di_daniele --name primo_container hello-world
```

```
podman pod restart pod_di_daniele
```

```
podman pod stop pod_di_daniele
```

```
podman pod kill pod_di_daniele
```

I POD

```
podman generate kube
```

```
dtrezza@SYS-DTREZZA:/mnt/c/Users/Daniele$ podman generate kube pod_di_daniele
# Generation of Kubernetes YAML is still under development!
#
# Save the output of this file and use kubectl create -f to import
# it into Kubernetes.
#
# Created with podman-3.3.1
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2021-11-09T15:25:43Z"
  labels:
    app: poddi daniel e
    name: pod_di_daniele
spec:
  containers:
  - command:
    - /hello
    env:
    - name: PATH
      value: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
    - name: TERM
      value: xterm
    - name: container
      value: podman
    image: docker.io/library/hello-world:latest
    name: primocontainer
    resources: {}
    securityContext:
      allowPrivilegeEscalation: true
      capabilities:
        drop:
        - CAP_MKNOD
        - CAP_NET_RAW
        - CAP_AUDIT_WRITE
      privileged: false
      readOnlyRootFilesystem: false
      selinuxOptions: {}
    workingDir: /
  dnsConfig: {}
  restartPolicy: Never
status: {}
```



Un progetto vero

Docker images and AWS lambda

```
FROM public.ecr.aws/lambda/python:3.9 as stage

RUN yum install -y -q sudo unzip

# Current stable version of Chromium
ENV CHROMIUM_VERSION=1002910

# Install Chromium
COPY install-browser.sh /tmp/
RUN /usr/bin/bash /tmp/install-browser.sh

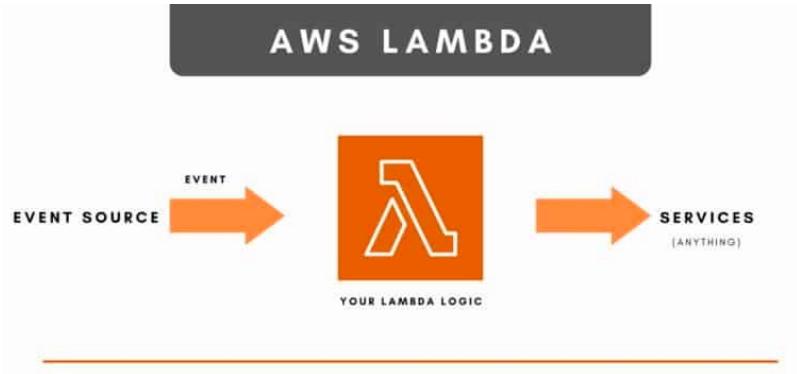
COPY chrome-deps.txt /tmp/
RUN yum install -y $(cat /tmp/chrome-deps.txt)

COPY requirements.txt /tmp/

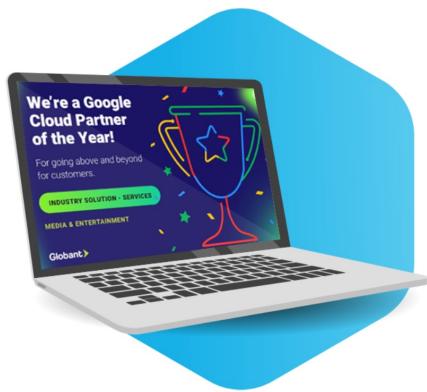
RUN python3 -m pip install --upgrade pip -q
RUN python3 -m pip install -r /tmp/requirements.txt -q

COPY --from=stage /opt/chrome /opt/chrome
COPY --from=stage /opt/chromedriver /opt/chromedriver
COPY . ${LAMBDA_TASK_ROOT}

CMD [ "run.handler" ]
```



Cloud Providers partnerships



Grazie per l'attenzione, *A presto!*

Daniele Trezza

daniele.trezza@globant.com

Nicholas Ferrari

nicholas.ferrari@globant.com