Bases de dades i Gestió de flotes de dispositius

Màster Internet of Things

25 i 27de Maig del 2020

Sessio 1
Introducció
Docker & BalenaOS

Qui soc?

David Raba
Enginyer Informàtic

Master en Business Innovation and Technology Management

CTO @ UBIKWA SYSTEMS

Backend Engineer - R&D - Full stack developer
Fliwer
Feb 2014 – Mar 2015 • 1 yr 2 mos
Barcelona Area, Spain

- Developing for a new start-up project: www.fliwer.com and www.fliwerpro.com
- Backend development: Artificial Intelligence and firmware development
- Data analysis and modeling layers
- Connected devices/Internet of Things/M2M
- C/C++ | Ruby | Apache | GIT | Puppet | Memcached | Gearman | Percona | HAProxy

R&D Analyst
Aplicaciones de Inteligencia Artificial (AIS - Barcelona)
Sep 2011 – Apr 2014 • 2 yrs 8 mos
Barcelona Area, Spain

- Data product creation
- Developing business reports, and data visualization
- Web scraping, Extract, manipulate, and transform data to draw actionable insights by using datamining and statistical modeling tools
- Prepare and deliver presentations to managers and executives
- Participate in ongoing decisions concerning data collection, study design, data analyses
- SAS | R | Ruby | Excel | Java

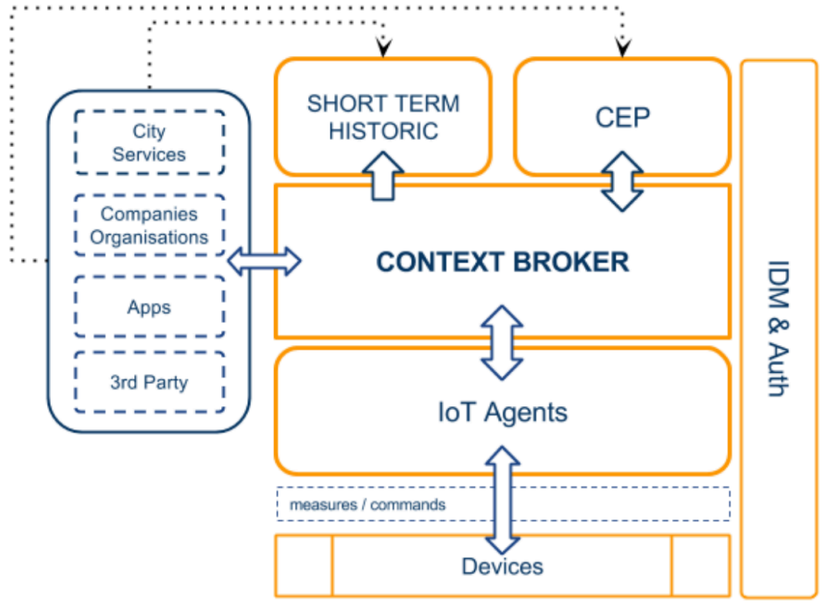
IT Architect / Developer
Netsuus Internet Intelligence
Jan 2009 – Aug 2011 • 2 yrs 8 mos

- Involved in all aspects of building and maintaining the production infrastructure and services
- Solid understanding of web analytics, e-commerce, and data analysis on clickstream fom online traffic collection.
- Core development of an online benchmarking system based on traffic sniffering
- Startup experience
- Ruby on Rails | Ruby | HTML | CSS | C++ | Pound | Apache

Start-ups

Development & Project Manager - Sporadically CTO
INSYLO TECHNOLOGIES SL
May 2015 – Present · 5 yrs 1 mo
Barcelona Area, Spain

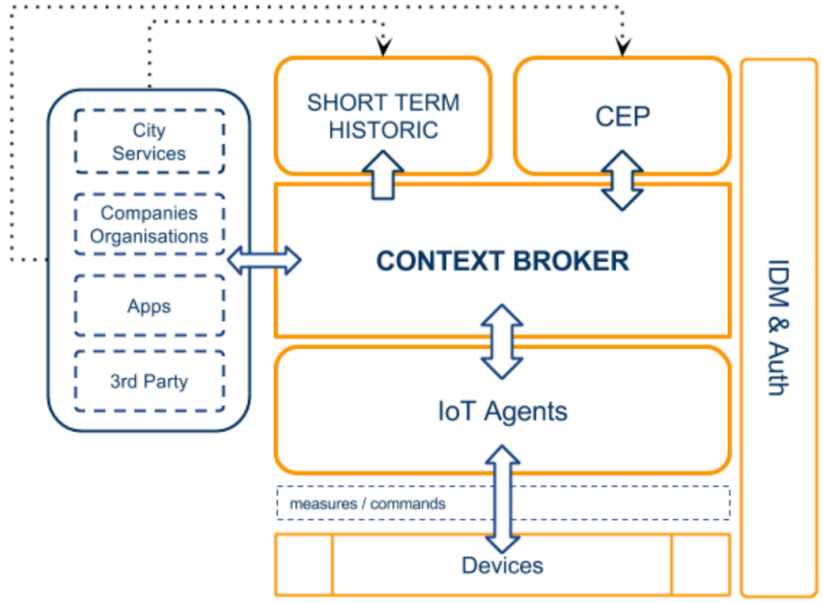
- To get a product out the door
- Deliver results to the market
- To make sure the development team is able to work as efficiently as possible and this means making sure they have clear goals
- From the initial project scope to deploying the product out to customer sites.
- And develop Android SDK | Golang | FIWARE | R | MongoDB | SQLServer | Python | Ruby | C | Docker | Azure | Git.

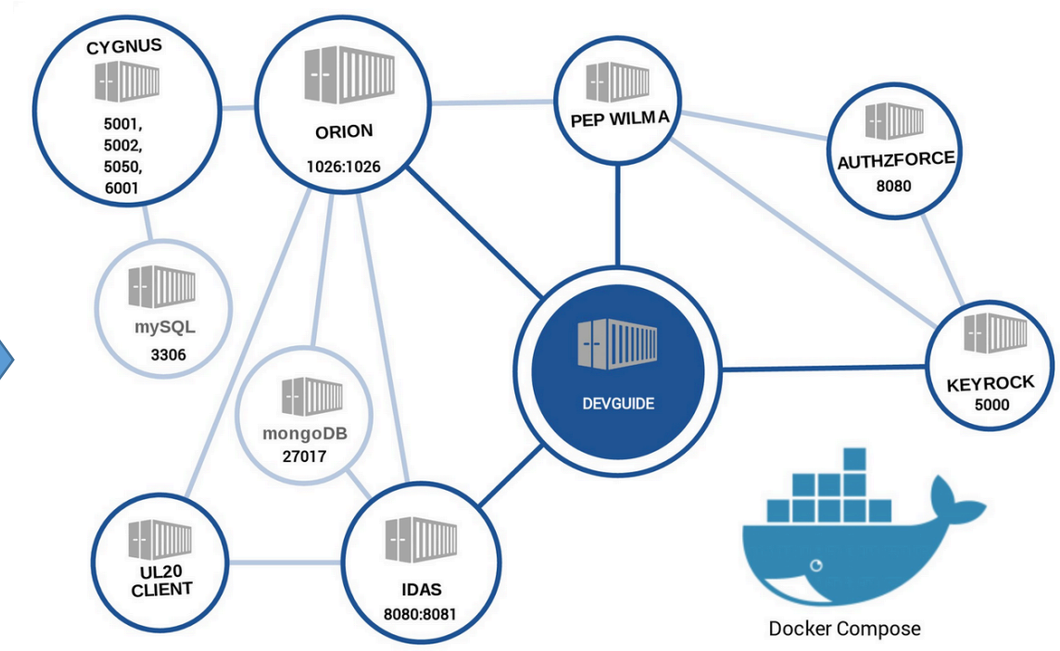


Start-ups

Development & Project Manager - Sporadically CTO
INSYLO TECHNOLOGIES SL
May 2015 – Present · 5 yrs 1 mo
Barcelona Area, Spain

- To get a product out the door
- Deliver results to the market
- To make sure the development team is able to work as efficiently as possible and this means making sure they have clear goals
- From the initial project scope to deploying the product out to customer sites.
- And develop Android SDK | Golang | FIWARE | R | MongoDB | SQLServer | Python | Ruby | C | Docker | Azure | Git.





Objectius

- **Sessió 1**
  - Docker
  - BalenaOS
  - Definició de serveis a través de Docker
  - Deploy de serveis a través de BalenaOS
- **Sessió 2**
  - Fonaments de Bases de Dades
  - Classificació de BBDD
- **Activitat**
  - Deploy d'un servei MQTT i un servei gateway contra una BBDD MongoDB (1a Sessió)
  - Deploy de un servei al dispositu de la RaspPi
    - Creació d'un tercer servei que reporti infrmació relevant via el servei local MQTT (A casa)
    - Realització del es consultes per explorer la BBDD generada. (A casa)

Sessió integrament Online
Activitats prèvies a la sesssió:
- Crear el compte personal a balenaCloud.
- Descarregar software robo3T (robomongo)
- Descarregar http://mqtt-explorer.com/
- Poder grabar la SD de la rasp durant la sessió

Cal revisar el material de BBDD prèviament a la sessió.
A la classe farem una repassada al material proporcionat, però la resta de la sessió consistirà en activitats sobre la base de dades onine.

Containers

Què son els containers?

- Abans dels standards dels containers

Què son els containers?

- I es van inventar els containers…

Al 1956 la majoria de vaixells eres carregats i descarregats a ma amb un cost de **$5.86** la tona.

Malcom McLean va neixer al 1913 i va desenvolupar el sistema de container intermodal, que va revolucionar el transport internacional.

El su principi deia que "*Un vaixell només guanya diners quan està navegant*"

Utilitzant containers, els seus costs eren de de **16 centims la tona.** La containerització va estandaritzar els processos de càrrega, reduint drásticament el temps i el cost aplicat.

A més, de dotar de fiabilitat al sistema.

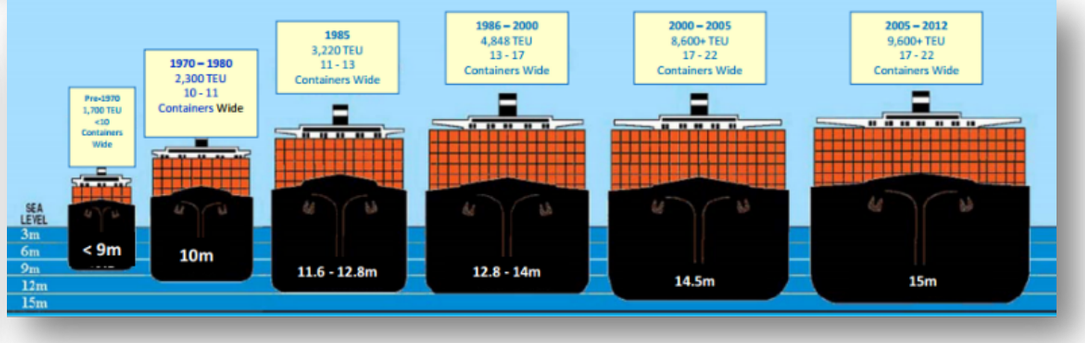
https://en.wikipedia.org/wiki/Malcom_McLean

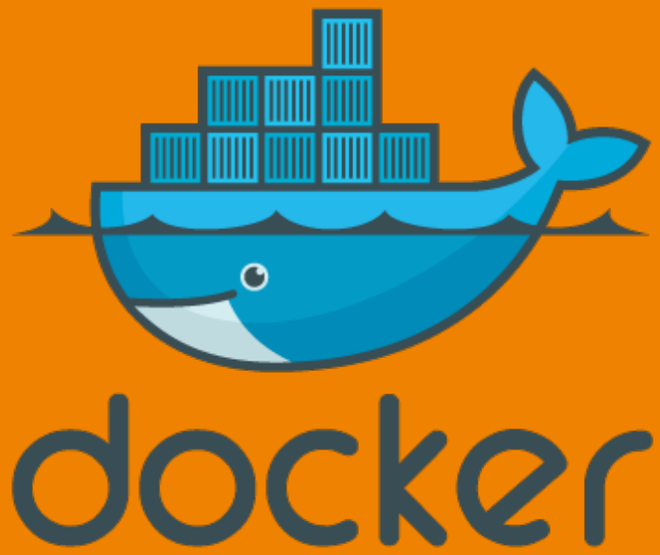
Què son els containers?

- Omplir el container

Què son els containers?

- Allotjat el container

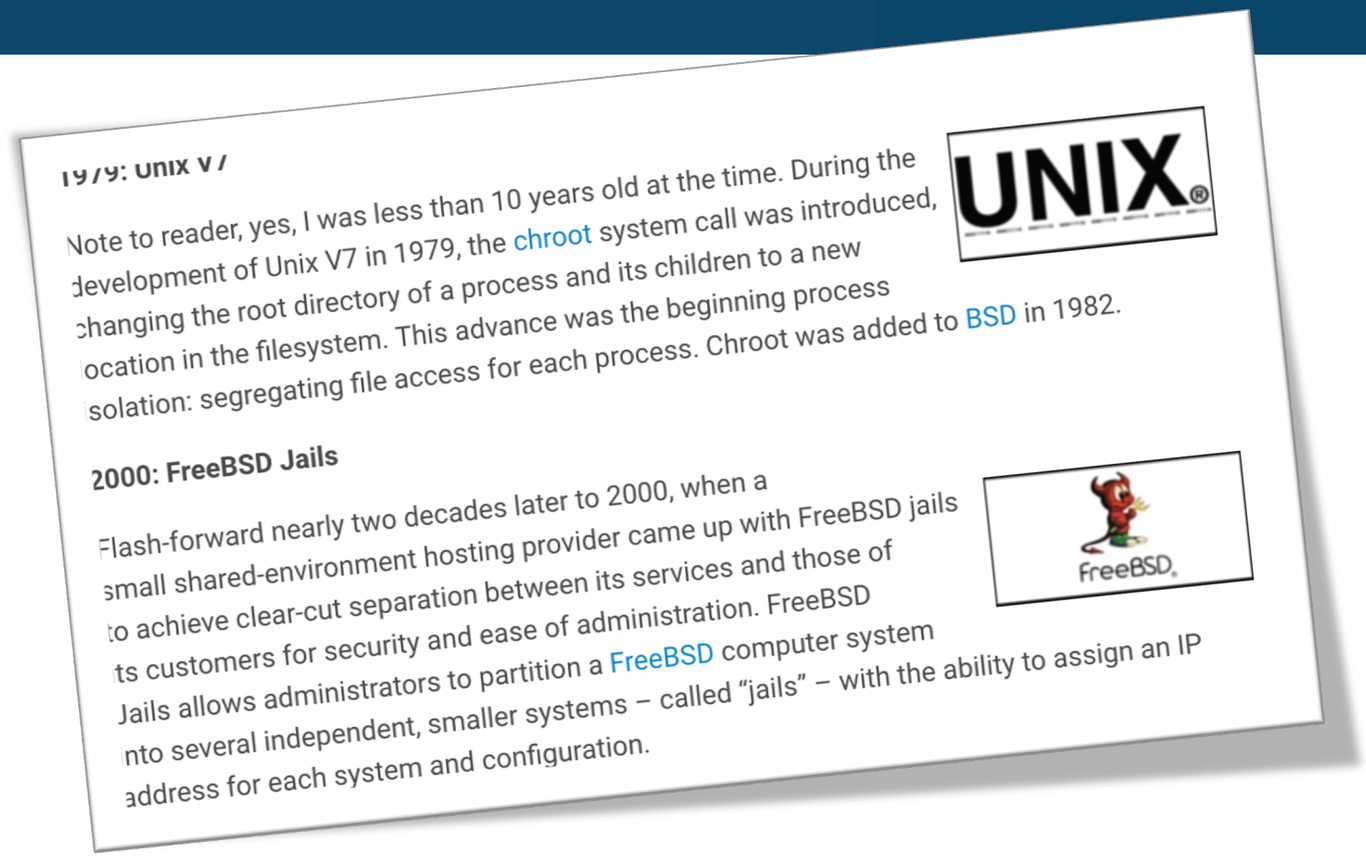



docker

Característiques de Docker

- Light-Weight
  - Mínima emprempta (*cpu/io/network*)
  - Basats en containers linux
  - Utilitza un sistema de fitxers multicapaper aprofitar espai (AUFS/LVM/OverlayFS)
  - Utilitza una estratègia copy-on-write per fer seguiment de canvis
- Portable
  - Pot funcionar en qualsevol Linux que doni suport a LXC.
  - La release 0.7 inclou suport per les distros RedHat/Fedora.
  - Raspberry pi support.
  - Planificades altres eines de distribució (lmctfy, etc.)
  - Planificats suports per altres sistemes operatius (Solaris, OSX, Windows)
- Autocontingut
  - Un contaoner Docker conté tot lo necessari per funcionar
  - És una base mínima del SO
  - Llibreries i frameworks
  - Codi d'aplicació
  - Un container docker ha de ser capaç de funcionar a qualsevol lloc on Docker pugui.

Historia



1979: Unix V7

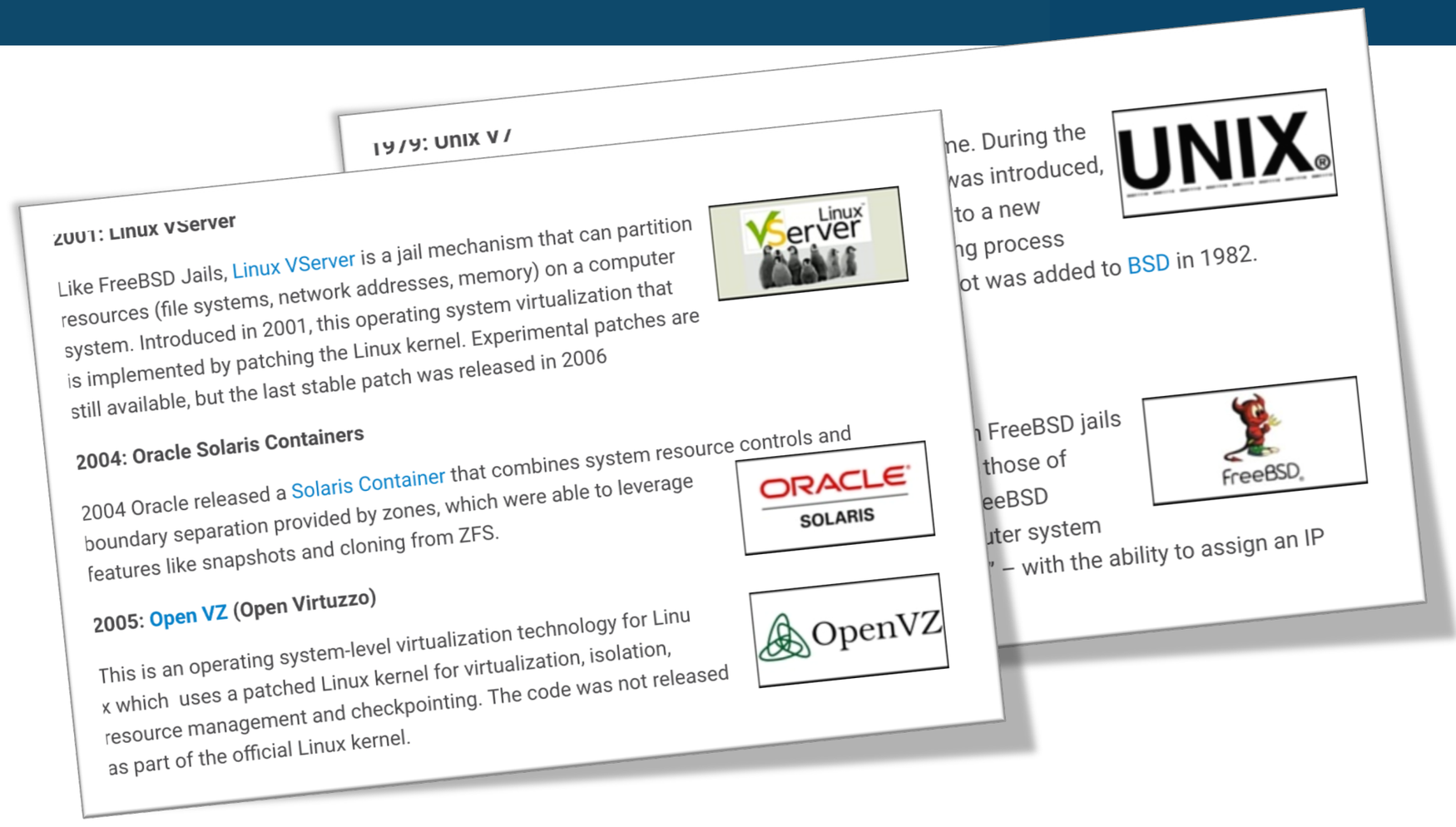
Note to reader, yes, I was less than 10 years old at the time. During the development of Unix V7 in 1979, the chroot system call was introduced, changing the root directory of a process and its children to a new location in the filesystem. This advance was the beginning process isolation: segregating file access for each process. Chroot was added to BSD in 1982.

2000: FreeBSD Jails

Flash-forward nearly two decades later to 2000, when a small shared-environment hosting provider came up with FreeBSD jails to achieve clear-cut separation between its services and those of its customers for security and ease of administration. FreeBSD Jails allows administrators to partition a FreeBSD computer system into several independent, smaller systems – called "jails" – with the ability to assign an IP address for each system and configuration.

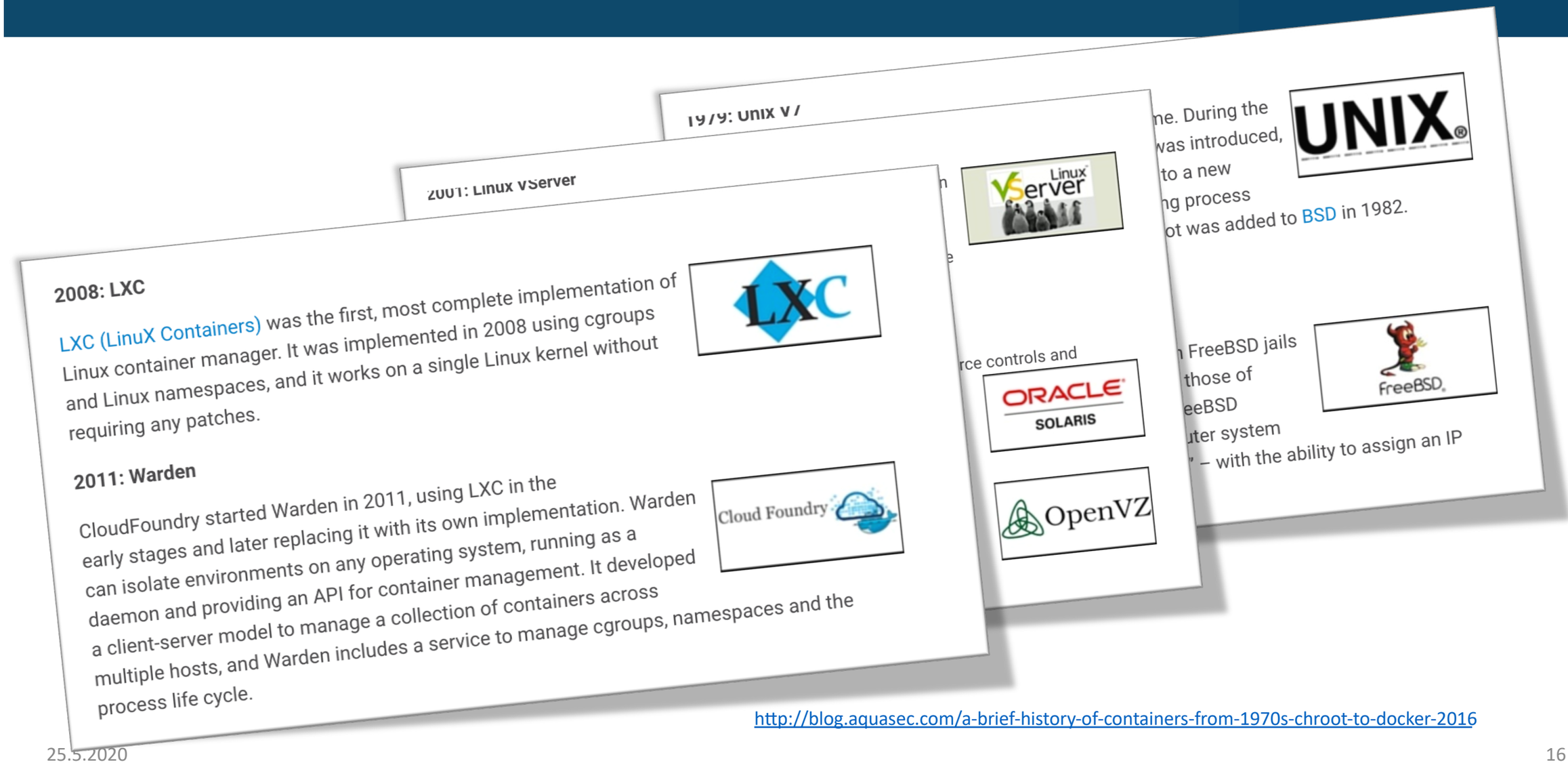
http://blog.aquasec.com/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016

Historia



http://blog.aquasec.com/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016

Historia



http://blog.aquasec.com/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016

Historia

Docker 0.1.0 released March 25, 2013

2013: Docker and the Future

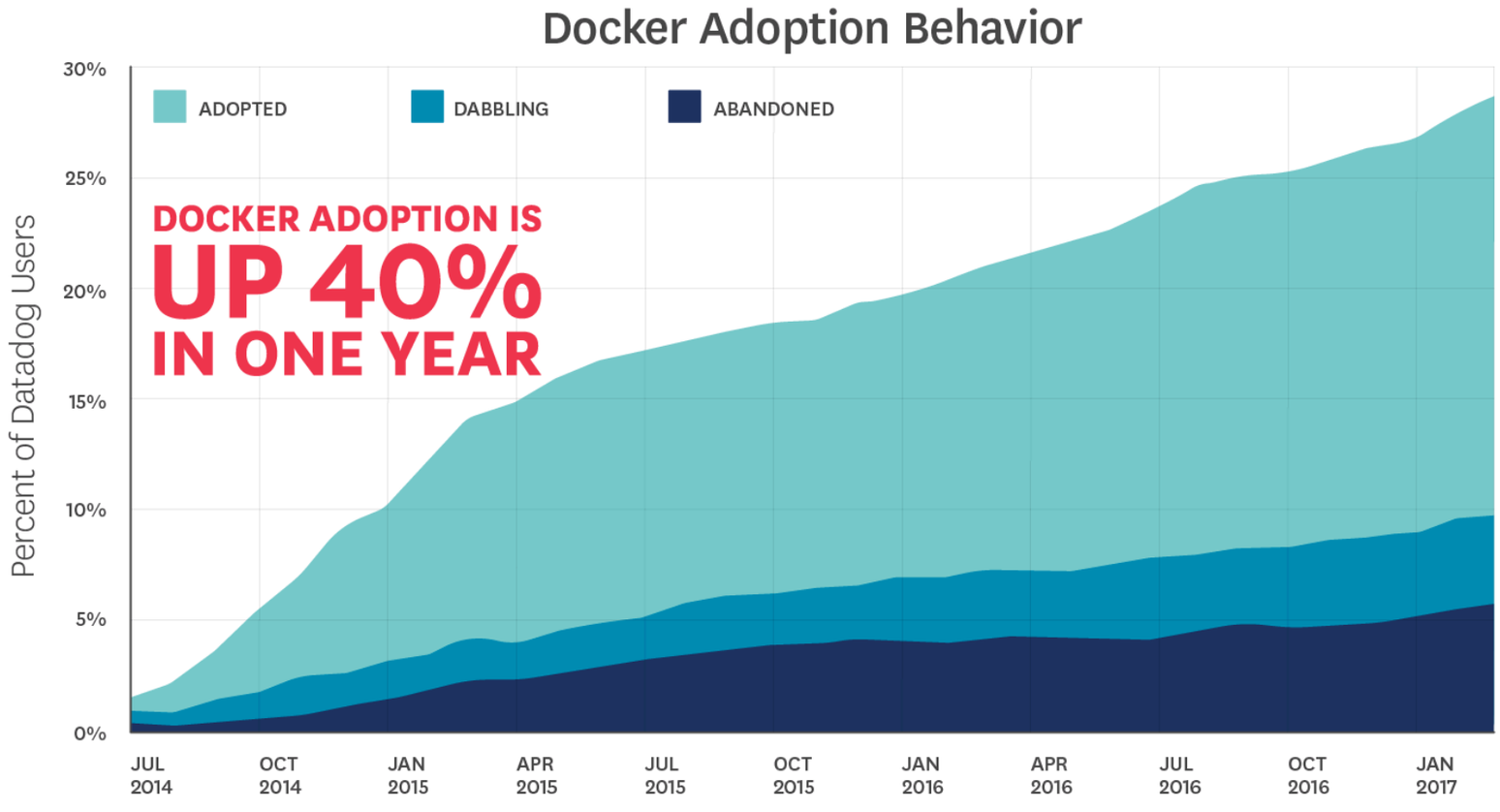
That's my (not so brief) summation of the pre-Docker container landscape. All those iterations had their adopters and devotees, but when Docker emerged in 2013, containers exploded in popularity. It's no coincidence the growth of Docker and container use goes hand-in-hand.

Just as Warden did, Docker also used LXC in its initial stages and later replaced that container manager with its own library, libcontainer. But I've no doubt that Docker separated itself from the pack by offering an entire ecosystem for container management.

With Docker, developers can create and run application containers quickly. And with the release of Docker Hub, developers can download and run application containers even faster.

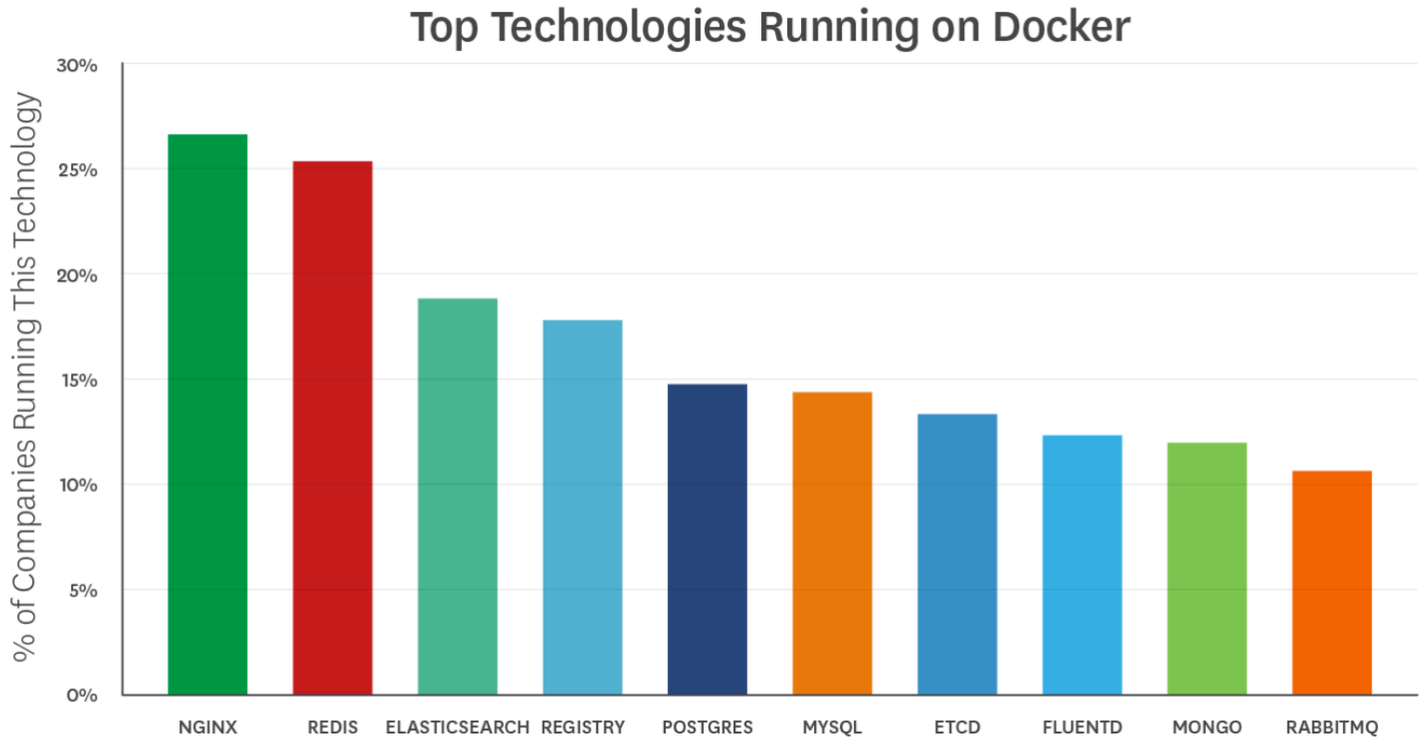
http://blog.aquasec.com/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016

Utilització de docker



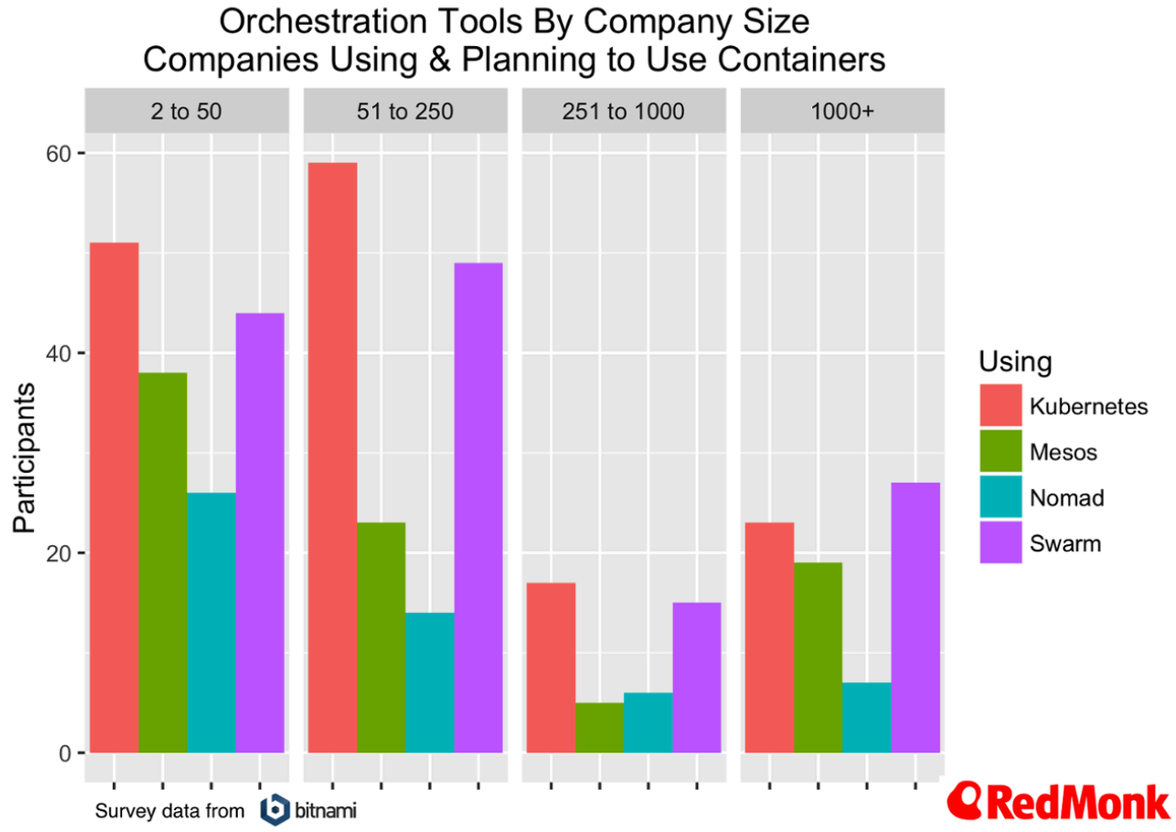
Source: Datadog

Utilització de docker



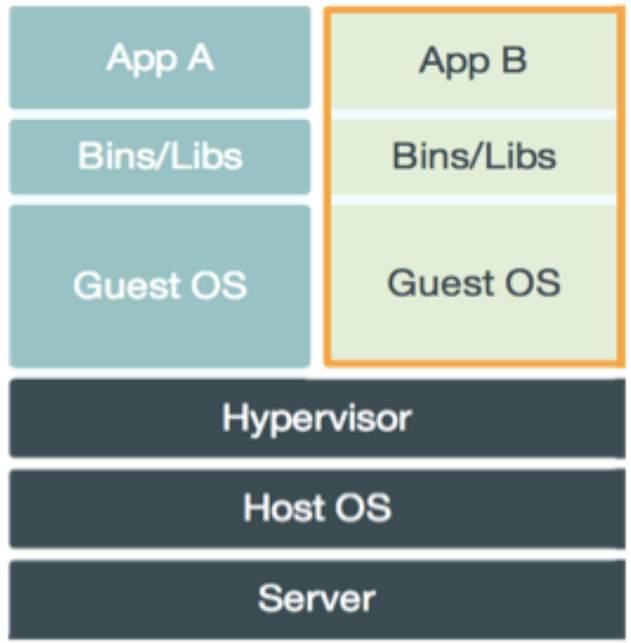
Source: Datadog

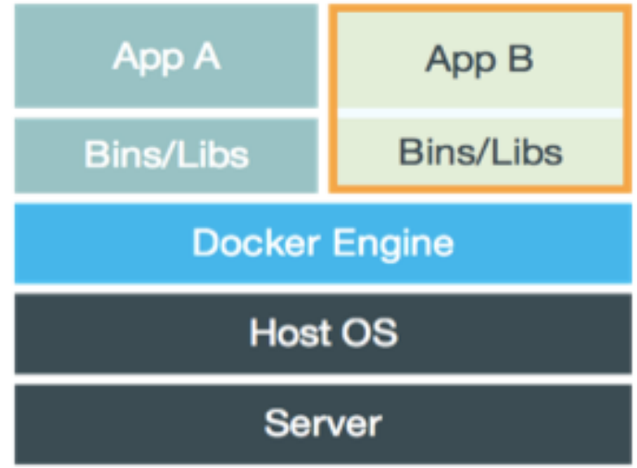
Utilització de docker



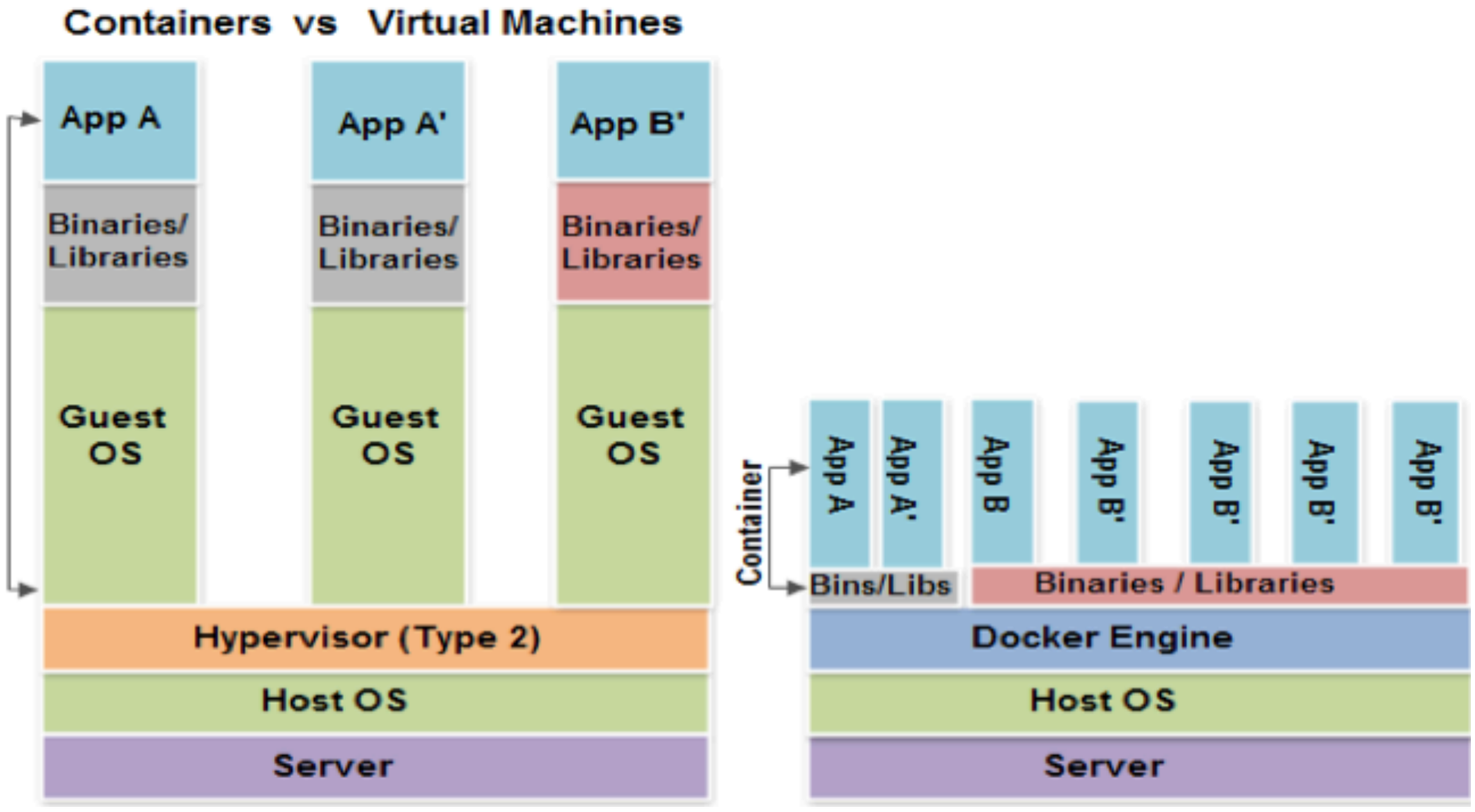
Docker vs Virtual Machines

- VM's
  - Instància completa del sistema Operatiu
  - No és fàcil la multi-instància
- Containers
  - Porcions del SO + components
  - Fàcil de duplicar, iniciar I aturar
  - OS Lleuger (Tiny Core Linux & Windows Server Core)





Docker vs Virtual Machines



Com de petits son els containers?

- ~24MB de descarrega
- S'executa complet a RAM
- Requirements mínims:
  - 46MB RAM
  - i486DX CPU (50MHz, 8KB cache)
- Requirements recomanats:
  - 128MB+ RAM
  - Pentium 2 CPU
- Ho tenim corrent amb una raspberry

Requeriments de host

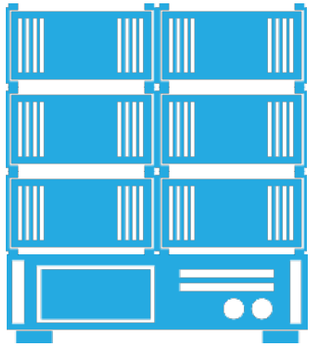
- Requirements mínims:
  - 256-512MB of RAM
  - 1GHz (x86) or 1.4Ghz (x64)CPU
- Recomanats
  - 512MB+
  - 2GHz+ CPU

On els podem posar?

Locally with:

- Docker Toolbox (Linux)
- Hyper V (Windows)

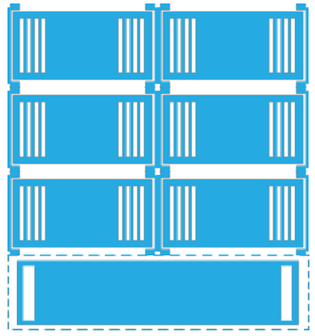
Locally



Physical Servers

- Linux (Linux)
- Windows 2016 TP3 (Windows)

On Premises



Clouds

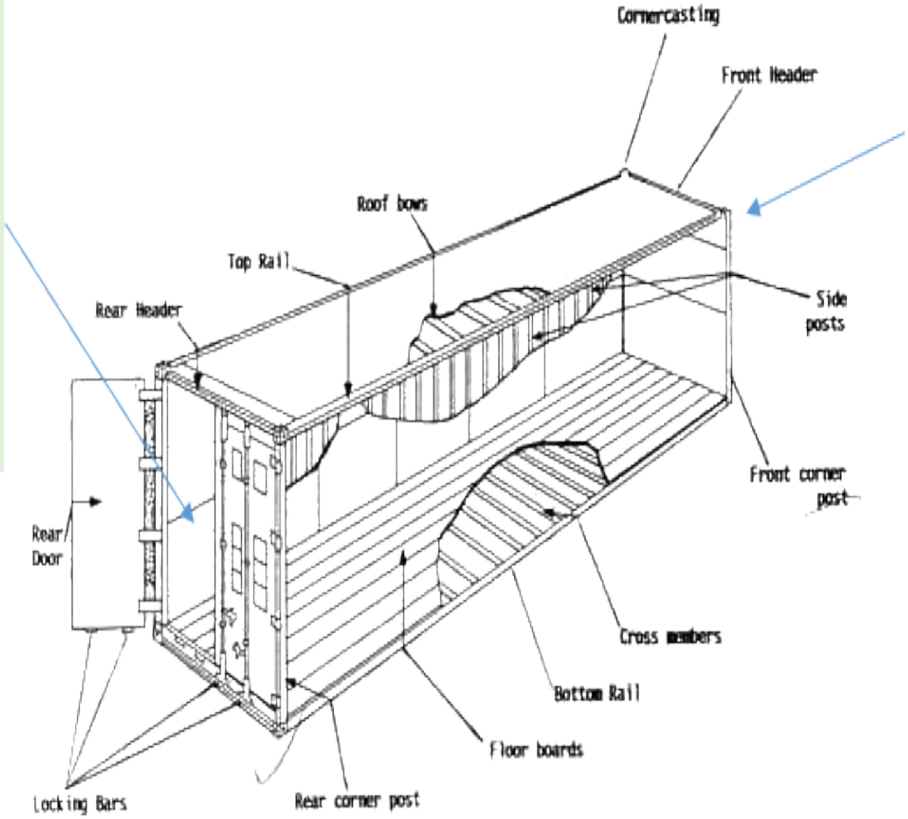
- Azure (Linux & Windows)
- Digital Ocean (Linux)
- AWS (Linux)
- Google (Linux)
- Rackspace (Linux)
- …etc.

Azure

Service Provider

Què en treiem de bo?

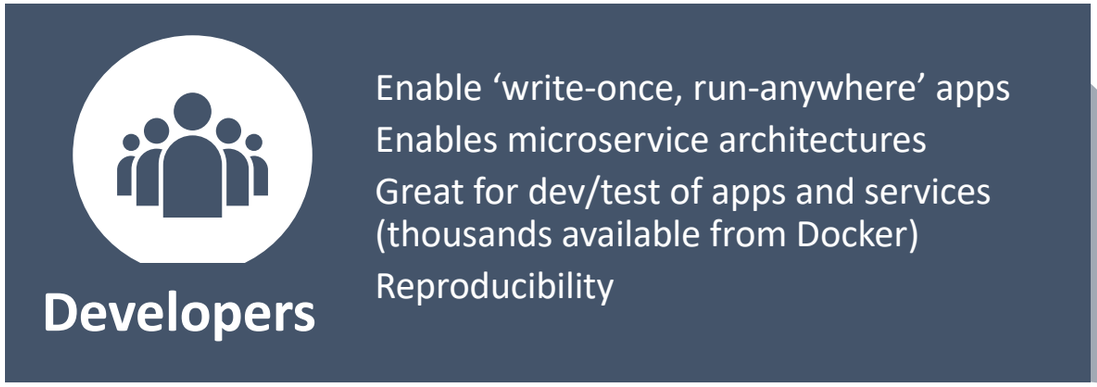
- Dan the Developer
  - Worries about what's "inside" the container
    - His code
    - His Libraries
    - His Package Manager
    - His Apps
    - His Data
  - All Linux servers look the same

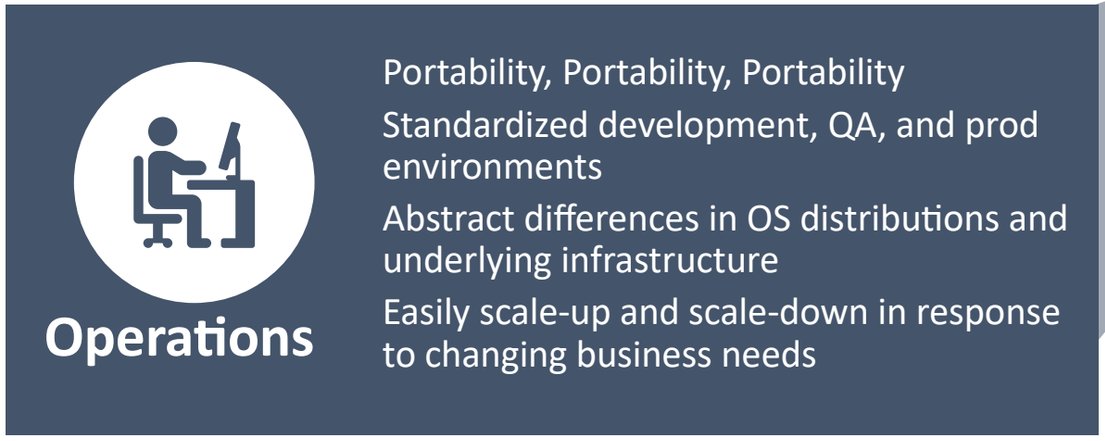


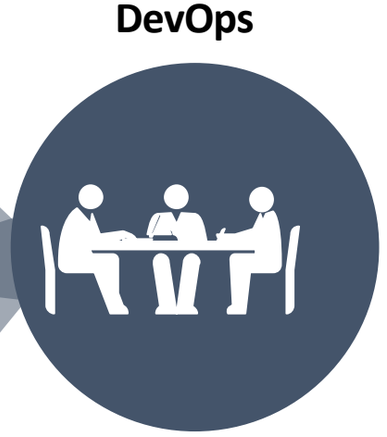
Major components of the container:

- Oscar the Ops Guy
  - Worries about what's "outside" the container
    - Logging
    - Remote access
    - Monitoring
    - Network config
  - All containers start, stop, copy, attach, migrate, etc. the same way

Què en treiem de bo?







Modern DevOps

1x every 2 weeks to 100x a day

MTTR 12x faster

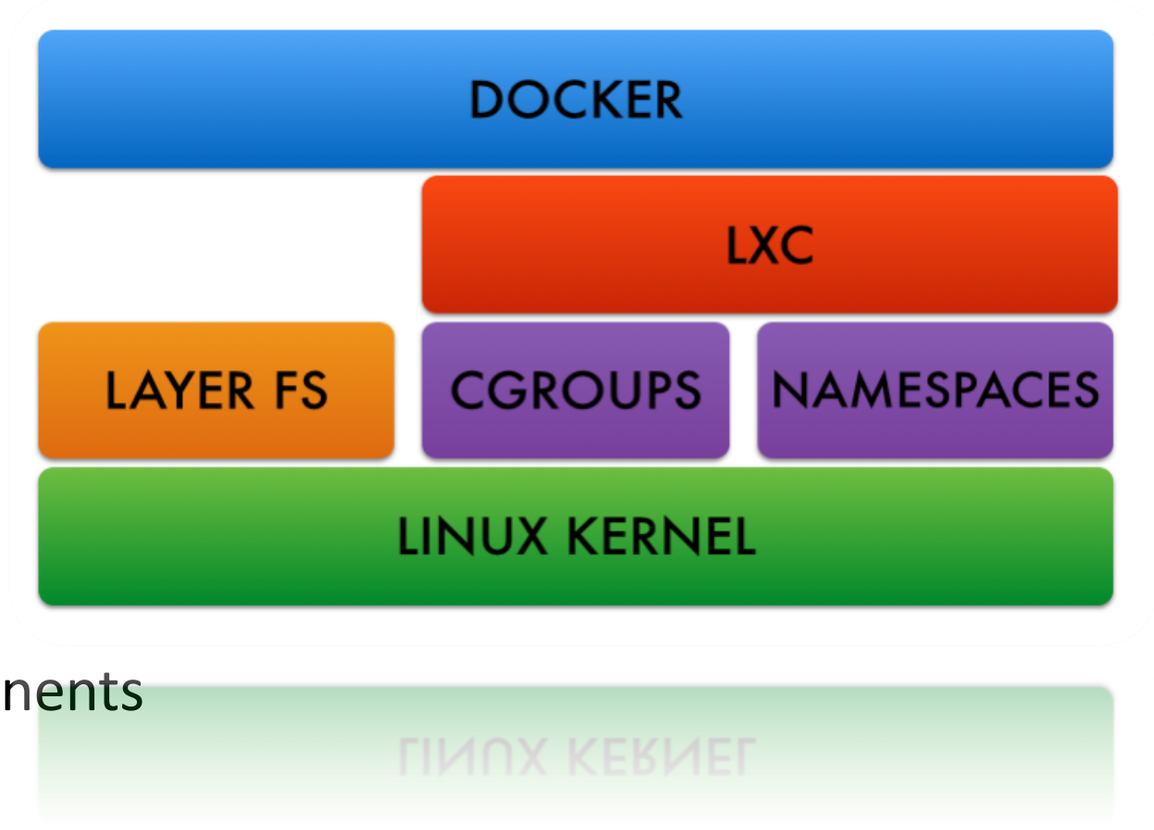
Amazon deploys every 11.6 seconds

Què en treiem de bo?

- La infraestructura es converteix en inmutable
- Inici ràpid
- Portable & lleuger
- Tenim una unitat de deploy
- Fàcil creació
- Cada container por ser una porció de la gran aplicació
  - Podem tenir múltiples containers que serveixes a més d'una aplicació (microserveis)

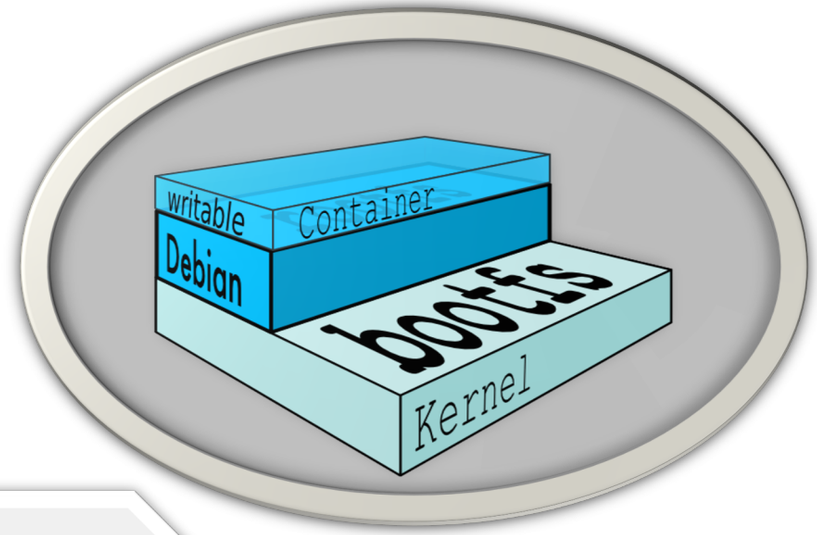
Terminologia

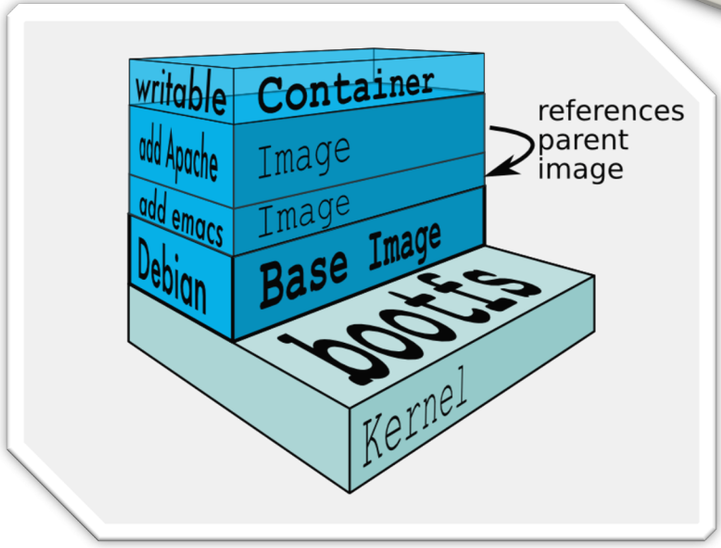
- Docker Engine
  - CLI
  - Docker Daemon
  - Docker Registry
- Docker Hub
  - Cloud service
    - Compartir Aplicacions
    - Automatitzar workflows
    - Composar apps a partir de components
- Docker images
- Docker containers



Docker images

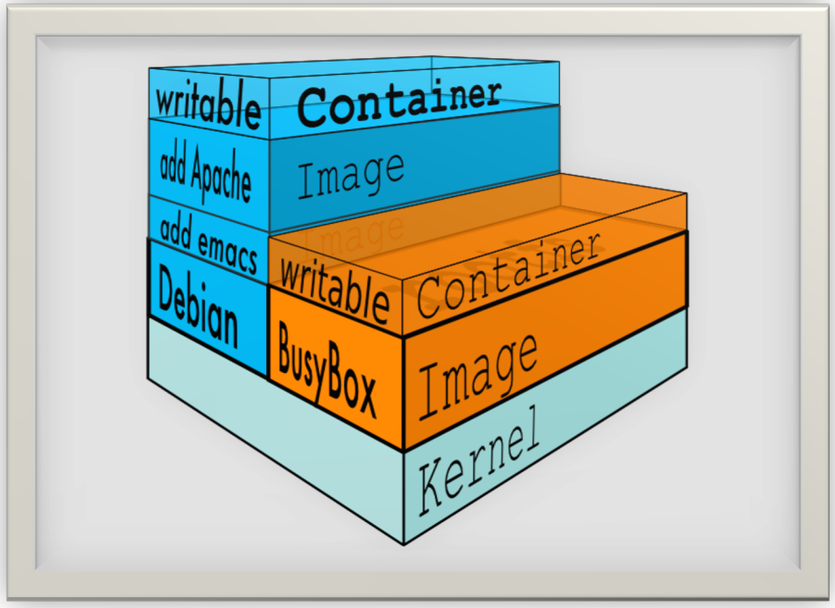
- No es un VHD
- No es un FILESYSTEM
- Fa ús de *Union File System*
- És read-only *Layer*
- No te estat
- Basicament un fitxer tar
- Te jerarquia
  - de profunditat arbitrària
- Pot encabir-se al Docker Registry





Docker container

És la unitat bàsica d'entrega de software (ship it!)



- S'executa a tot arreu
  - independentment de la versió de kernel
  - o de la distro del host
  - * però han de coincidir amb arquitectura

(x32, x64, etc)

- Ho executa tot
  - si pot correr a un host, ho pot fer dins d'un container

*A menys que es facin servir emuladors de CPU amb **qemu i binfmt**

Docker CLI

```
$ docker images      # mostra tots els images.
$ docker import      # crea una imatge des d'un arxiu tar.
$ docker build       # crea una imatge de d'un Dockerfile.
$ docker commit      # crea una image des de un container.
$ docker rmi         # elimina una image.
$ docker history     # llista els canvis d'una imatge.
```

Cicle de vida d'un container

La vida d'un container…

- Concepció
  - **BUILD** una imatge a partir d'un Dockerfile
- Neixement
  - **RUN** (create+start) un container
- Reprocció
  - **COMMIT** (persisteix) un container a una imatge
  - **RUN** un nou container des d'una imatge
- Sleep
  - **KILL** un continer en execució
- Wake
  - **START** un container aturat
- Death
  - **RM** (delete) un container aturat
- Extinció
  - **RMI** una imatge de container (delete image)

Dockerfile

- Conceptualment és un Makefile
- Extend una imatge base
- Que es converteix en una nova imatge
- Imperatiu, no Declaratiu

Defineix la recepta per construir una imatge

- S'utilitza docker build per executar un dokerfile
- Es poden definir ordres per defecte per executar, definir ports exposats, etc.

```
# our base image
FROM alpine:latest

# Install python and pip
RUN apk add --update py-pip

# upgrade pip
RUN pip install --upgrade pip

# install Python modules needed by the Python app
COPY requirements.txt /usr/src/app/
RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt

# copy files required for the app to run
COPY app.py /usr/src/app/
COPY templates/index.html /usr/src/app/templates/

# tell the port number the container should expose
EXPOSE 5000

# run the application
CMD ["python", "/usr/src/app/app.py"]
Macbook:flask-app draba$ cat Dockerfile
# our base image
FROM alpine:latest

# Install python and pip
RUN apk add --update py-pip

# upgrade pip
RUN pip install --upgrade pip

# install Python modules needed by the Python app
COPY requirements.txt /usr/src/app/
RUN pip install --no-cache-dir -r /usr/src/app/requirements.txt
```

Dockerfile: docker-run-vs-cmd-vs-entrypoint

In a nutshell

- RUN executes command(s) in a new layer and creates a new image. E.g., it is often used for installing software packages.
- CMD sets default command and/or parameters, which can be overwritten from command line when docker container runs.
- ENTRYPOINT configures a container that will run as an executable.

```
RUN apt-get install python3
CMD echo "Hello world"
ENTRYPOINT echo "Hello world"
```

When instruction is executed in *shell* form it calls `/bin/sh -c <command>` under the hood and normal shell processing happens. For example, the following snippet in Dockerfile

```
ENV name John Dow
ENTRYPOINT echo "Hello, $name"
```

when container runs as `docker run -it <image>` will produce output

```
Hello, John Dow
```

```
ENTRYPOINT ["/bin/echo", "Hello"]
CMD ["world"]
```

when container runs as `docker run -it <image>` will produce output

```
Hello world
```

but when container runs as `docker run -it <image> John` will result in

```
Hello John
```

https://goinbigdata.com/docker-run-vs-cmd-vs-entrypoint/

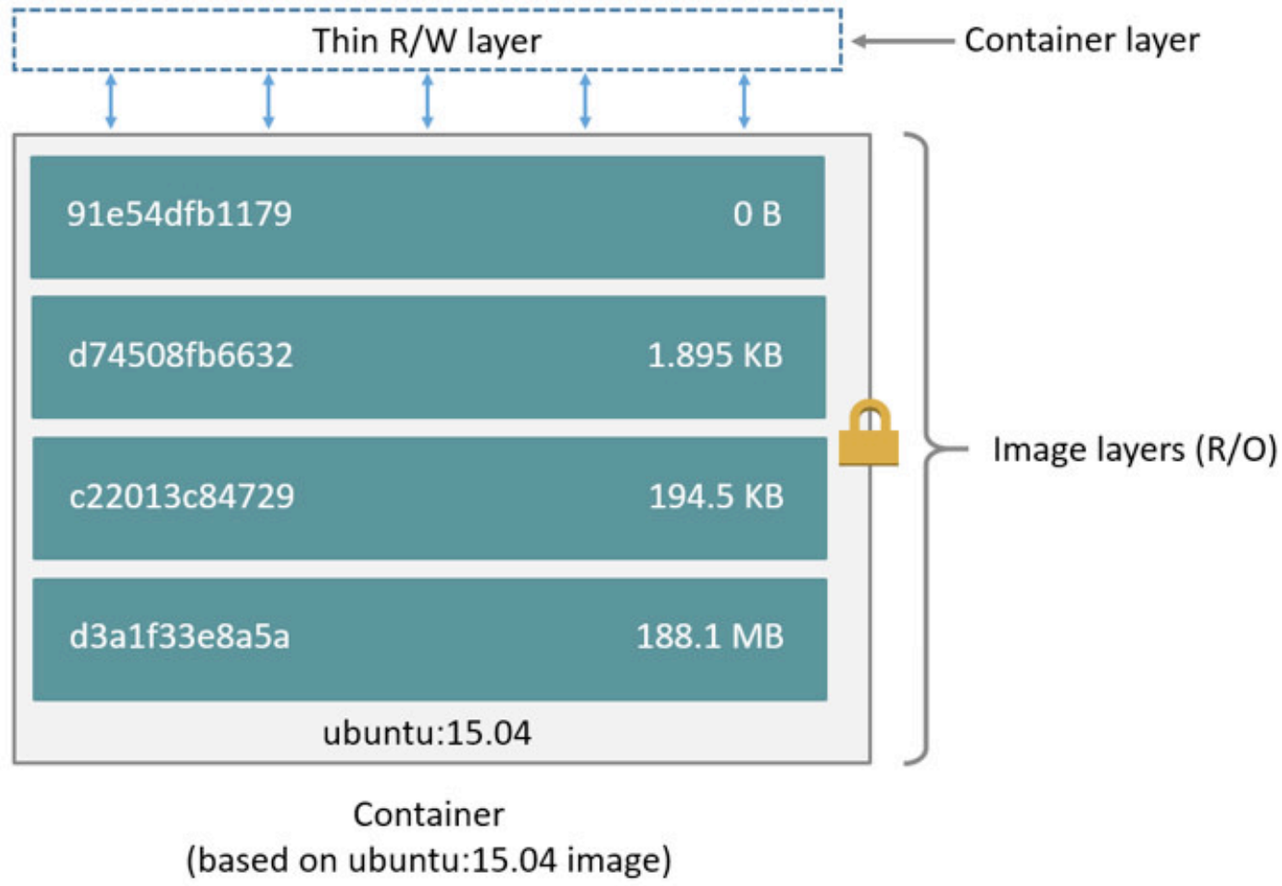
Dockerfile: RUN

```
FROM ubuntu:18.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

acme/my-base-image:1.0

```
FROM ubuntu:18.04
COPY . /app
```

```
FROM acme/my-base-image:1.0
CMD /app/hello.sh
```



5 preguntes típiques sobre Docker

1. Docker client versus host
   - Docker client is a command line interface (CLI) Docker
   - Docker host is a Linux/Windows VM running Docker daemon
2. Docker Linux and Windows hosts
   - You can only create the same container as the underlying host VM - Linux host = Linux containers
3. Docker Image vs Docker Container
   - Image = The definition – literally a single file [My Website]
   - Container – An instance of an image [3 instances of My Website]
4. Cloud Registry Service and Public Image Repos
   - Unlimited public repos, one free private or buy private repos
   - 50,000+ images - Wordpress, Nginx, Redis, MySQL, Logstash, and your images!
   - Docker Trusted Registry – Dedicated registry application deployable on-premise or direct from Azure Marketplace

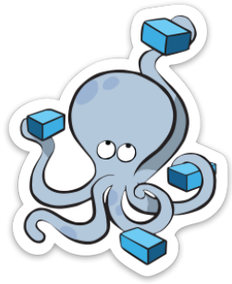
5 preguntes típiques sobre Docker

5. Deployments replace instead of update

"Immutable infrastructure"

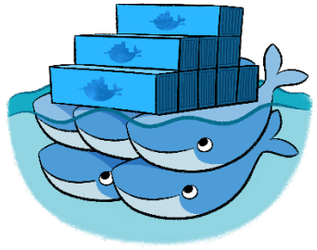
Website	**Docker**
Update your app using Web Deploy or CI/CD	Replace running containers using CI, Don't update the old container

Altres conceptes Docker



Compose

Definir i fer deploy d'aplicacions multicontainer



Swarm

Ús de multiples màquines com a una de sola, per controlar multiples entorns de containers.

Docker Machine

Crear i gestionar instàncies Docker localment i en cloud

Casos

```
FROM frolvlad/alpine-glibc:latest

ENV REFRESHED_AT 2017-12-01

# Set up a tools dev directory
WORKDIR /home/dev

RUN apk add --no-cache \
    git \
    python \
    scons \
    && apk --update --no-cache add --virtual build-dependencies \
    bzip2-dev \
    ca-certificates \
    openssl \
    tar \
    w3m \
    && wget https://developer.arm.com/-/media/Files/downloads/gnu-rm/6-2017q2/gc
c-arm-none-eabi-6-2017-q2-update-linux.tar.bz2 \
    && tar xvf gcc-arm-none-eabi-6-2017-q2-update-linux.tar.bz2 \
    && rm gcc-arm-none-eabi-6-2017-q2-update-linux.tar.bz2 \
    && apk del build-dependencies \
    && rm -rf /home/dev/gcc-arm-none-eabi-6-2017-q2-update/share/doc

# Set up the compiler path
ENV PATH="/home/dev/gcc-arm-none-eabi-6-2017-q2-update/bin:${PATH}"

WORKDIR /usr/project

CMD ["scons"]
```

```
$ docker build -t="feabhas/gcc-arm-scons-alpine:1.0" .

$ cd ../blog-test-project
$ docker run --rm -v $(pwd):/usr/project feabhas/gcc-arm-scons-alpine:1.0

$ docker push feabhas/gcc-arm-scons-alpine:1.0
```

Finally we can update our test project's `bitbucket-pipelines.yml` file to build with our new, leaner, Docker image:

```
image: feabhas/gcc-arm-scons-alpine:1.0

pipelines:
  default:
    - step:
        script:
          - scons
```

https://blog.feabhas.com/2017/12/introduction-docker-embedded-developers-part-4-reducing-docker-image-size/

Casos

```
FROM nginx:1.13.3

RUN rm /etc/nginx/nginx.conf
COPY nginx.conf /etc/nginx/

RUN rm /etc/nginx/conf.d/default.conf
COPY project.conf /etc/nginx/conf.d/

COPY ./static ./static
```

```
version: '3'

services:

  dash_app:
    container_name: dash_app
    restart: always
    build: ./dash_app
    ports:
      - "7501:7501"
    command: gunicorn -w 1 --timeout 200 -b :7501 app:server
    volumes:
      - /root/api/docker/service/api/backup:/data
      - /root/apps/volume-estimation-rest/ubk-volum-estimator/rest-server/app/tmp/:/processing
    links:
      - mongo

  mongo:
    restart: always
    build: ./mongotunnel
    ports:
      - "37017:37017"

  nginx:
    container_name: nginx
    restart: always
    build: ./nginx
    ports:
      - "7500:7500"
    depends_on:
      - dash_app
```

Casos


```
FROM python:3.6.2

RUN mkdir -p /data
VOLUME /data

## Prepare apt-get:
RUN echo "debconf debconf/frontend select Noninteractive" | debconf-set-selections && \
    apt-get update -qy && \
    apt-get clean autoclean && \
    apt-get autoremove -y && \
    apt-get install git -qy && \
    apt-get -y install openssh-client -qy && \
    apt-get clean

# Set up app dependencies and get bitbucket applications
# Make ssh dir
RUN mkdir /root/.ssh/

# Copy over private key, and set permissions
COPY ./config/ssh/config /root/.ssh/
ADD ./config/ssh/id_bitbucket /root/.ssh/id_bitbucket
RUN chmod 0600 "/root/.ssh/id_bitbucket"

#Install ssh tools
RUN apt-get -yqq install ssh
RUN ssh-keyscan -T 60 bitbucket.org >> /root/.ssh/known_hosts

# Add bitbuckets key
RUN ssh-keyscan bitbucket.org >> /root/.ssh/known_hosts

WORKDIR /tmp
RUN git clone git@bitbucket.org:ubikwa/reference-depthmap-estimation.git
WORKDIR /tmp/reference-depthmap-estimation
RUN pip install --no-cache-dir --upgrade -r requirements.txt .

RUN mkdir -p /app
WORKDIR /app
COPY requirements.txt /app
RUN pip install --no-cache-dir -r requirements.txt
```

```
version: '3'

services:

  dash_app:
    container_name: dash_app
    restart: always
    build: ./dash_app
    ports:
      - "7501:7501"
    command: gunicorn -w 1 --timeout 200 -b :7501 app:server
    volumes:
      - /root/api/docker/service/api/backup:/data
      - /root/apps/volume-estimation-rest/ubk-volum-estimator/rest-server/app/tmp/:/processing
    links:
      - mongo

  mongo:
    restart: always
    build: ./mongotunnel
    ports:
      - "37017:37017"

  nginx:
    container_name: nginx
    restart: always
    build: ./nginx
    ports:
      - "7500:7500"
    depends_on:
      - dash_app
```

Casos

```
RUN mkdir -p /app
WORKDIR /app
COPY requirements.txt /app
RUN pip install --no-cache-dir -r requirements.txt

COPY . /app

VOLUME /app
```

```
version: '3'

services:

  dash_app:
    container_name: dash_app
    restart: always
    build: ./dash_app
    ports:
      - "7501:7501"
    command: gunicorn -w 1 --timeout 200 -b :7501 app:server
    volumes:
      - /root/api/docker/service/api/backup:/data
      - /root/apps/volume-estimation-rest/ubk-volum-estimator/rest-server/app/tmp/:/processing
    links:
      - mongo

  mongo:
    restart: always
    build: ./mongotunnel
    ports:
      - "37017:37017"

  nginx:
    container_name: nginx
    restart: always
    build: ./nginx
    ports:
      - "7500:7500"
    depends_on:
      - dash_app
```

Cas


```
# MongoDB Dockerfile
#

# Pull base image.
FROM ubuntu:14.04

ENV DEBIAN_FRONTEND noninteractive

# make sure the package repository is up to date and update ubuntu
RUN \
  sed -i 's/# \(.*multiverse$\)/\1/g' /etc/apt/sources.list && \
  apt-get update && \
  apt-get -y upgrade && \
  locale-gen en_US.UTF-8
RUN apt-get install -y curl htop man software-properties-common unzip vim wget

ENV LANG en_US.UTF-8
ENV LANGUAGE en_US.UTF-8
ENV LC_ALL en_US.UTF-8
ENV HOME /root

# supervisor installation &&
# create directory for child images to store configuration in
RUN apt-get -y install supervisor && \
  mkdir -p /var/log/supervisor && \
  mkdir -p /etc/supervisor/conf.d

# supervisor base configuration
ADD supervisor.conf /etc/supervisor.conf
# default command

RUN wget https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-3.2.9.tgz -O /tmp/mongo.tar.gz
RUN mkdir -p /data/db
RUN ln -s /opt/mongodb/bin/mongo /usr/local/bin/mongo
RUN ln -s /opt/mongodb/bin/mongod /usr/local/bin/mongod
RUN (cd /tmp && tar zxf mongo.tar.gz && mv mongodb-* /opt/mongodb)
RUN rm -rf /tmp/*
```

```
ADD mongo.sv.conf /etc/supervisor/conf.d/

# Define mountable directories.
VOLUME ["/data"]
VOLUME ["/data/db"]

# Define working directory.
WORKDIR /data

# Define default command.
CMD ["mongod"]

# Expose ports.
#   - 27017: process
#   - 28017: http
EXPOSE 27017
EXPOSE 28017

ADD etc/ /etc/
ADD supervisor.conf /etc/supervisor.conf
CMD ["supervisord", "-c", "/etc/supervisor.conf"]
```

Create, Run, etc...

docker build -t company/mongo .

docker run --name mongodb -d -v /data:/data -v /data/db:/data/db -p 27017:27017 -p 28017:28017 company/mongo

root@localhost:~/apps/masteriot# docker-compose logs -f

```
Attaching to masteriot_mongo_1
mongo_1  | 2020-05-19 06:56:35,295 CRIT Supervisor running as root (no user in config file)
mongo_1  | 2020-05-19 06:56:35,295 WARN Included extra file "/etc/supervisor/conf.d/mongo.sv.conf" during parsing
mongo_1  | 2020-05-19 06:56:35,301 INFO supervisord started with pid 1
mongo_1  | 2020-05-19 06:56:36,303 INFO spawned: 'mongo' with pid 9
mongo_1  | 2020-05-19 06:56:37,679 INFO success: mongo entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)
mongo_1  | 2020-05-19 06:56:42,851 WARN received SIGTERM indicating exit request
mongo_1  | 2020-05-19 06:56:42,851 INFO waiting for mongo to die
mongo_1  | 2020-05-19 06:56:42,978 INFO stopped: mongo (exit status 0)
mongo_1  | 2020-05-19 06:56:49,840 CRIT Supervisor running as root (no user in config file)
mongo_1  | 2020-05-19 06:56:49,840 WARN Included extra file "/etc/supervisor/conf.d/mongo.sv.conf" during parsing
mongo_1  | 2020-05-19 06:56:49,843 INFO supervisord started with pid 1
mongo_1  | 2020-05-19 06:56:50,846 INFO spawned: 'mongo' with pid 9
mongo_1  | 2020-05-19 06:56:52,323 INFO success: mongo entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)
^CERROR: Aborting.
root@localhost:~/apps/masteriot#
```

balenaOS

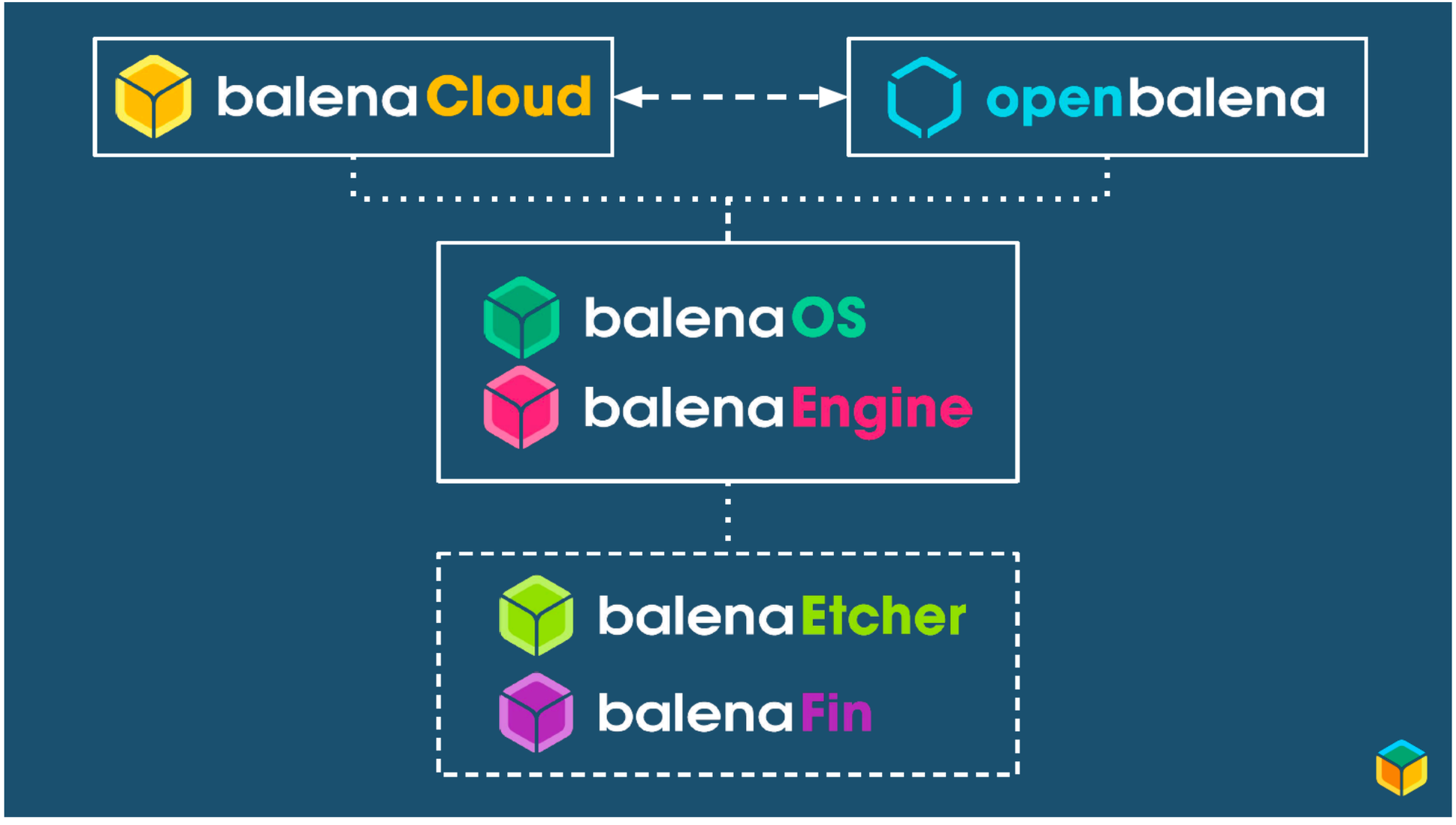
Docs Getting started Download

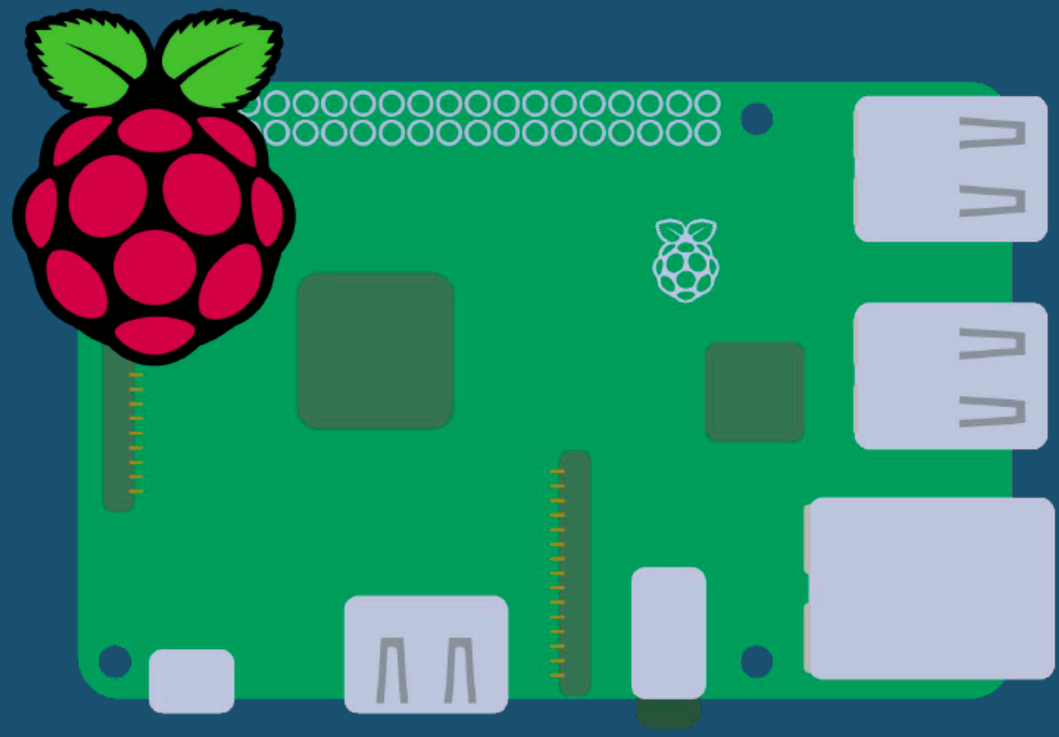
Run Docker containers on embedded devices

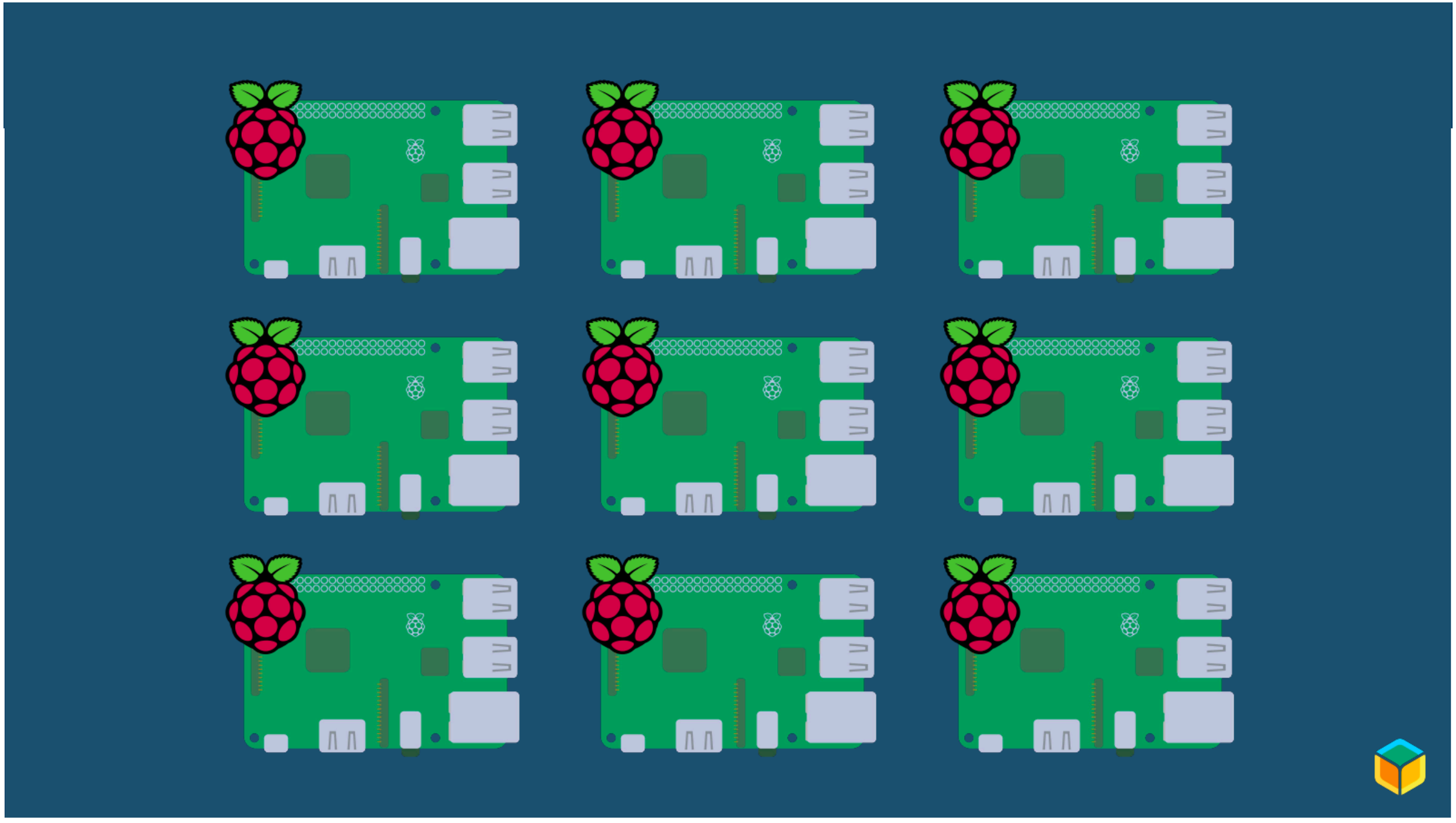
A host OS tailored for containers, designed for reliability, proven in production.

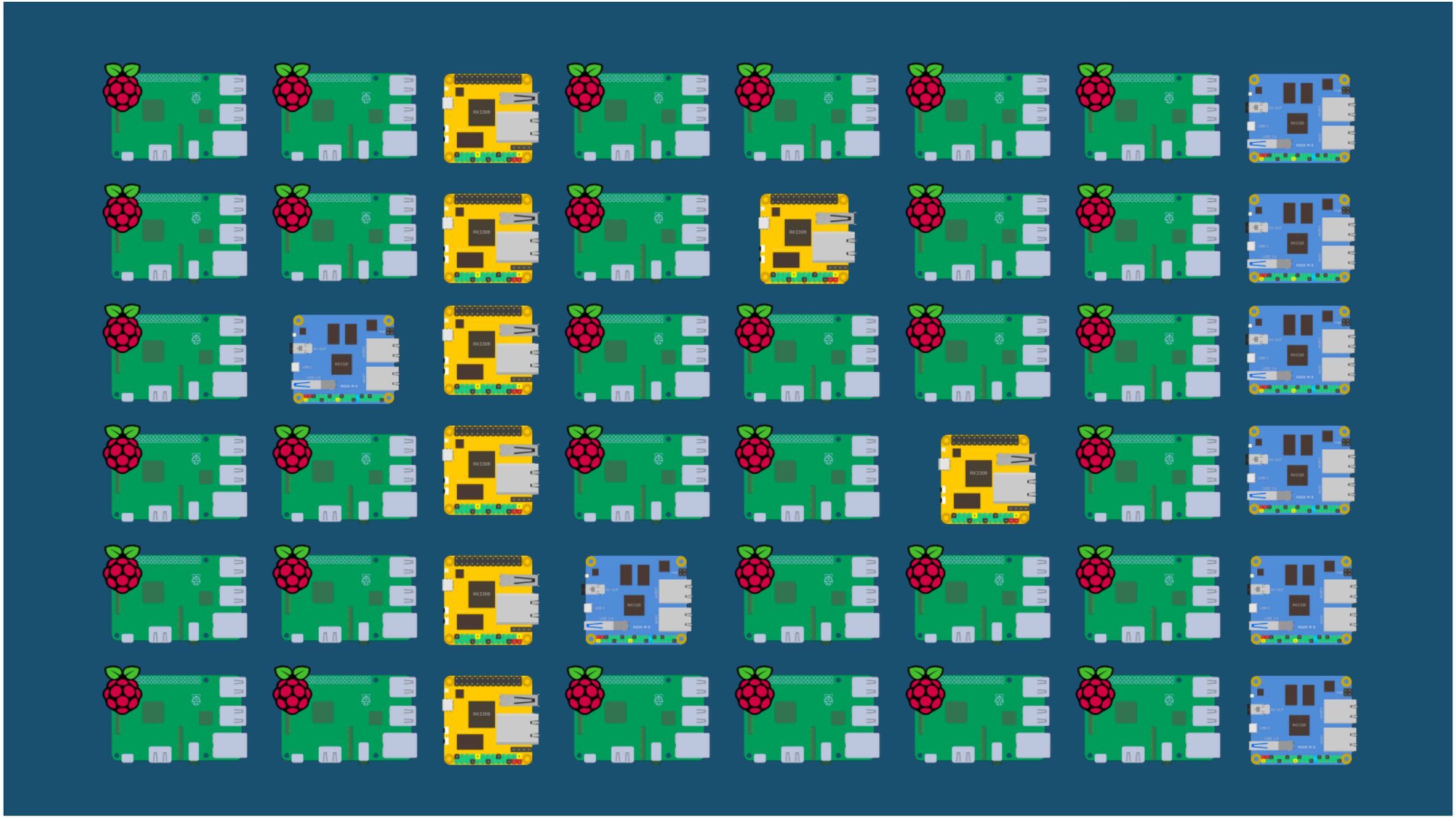
Try balenaOS

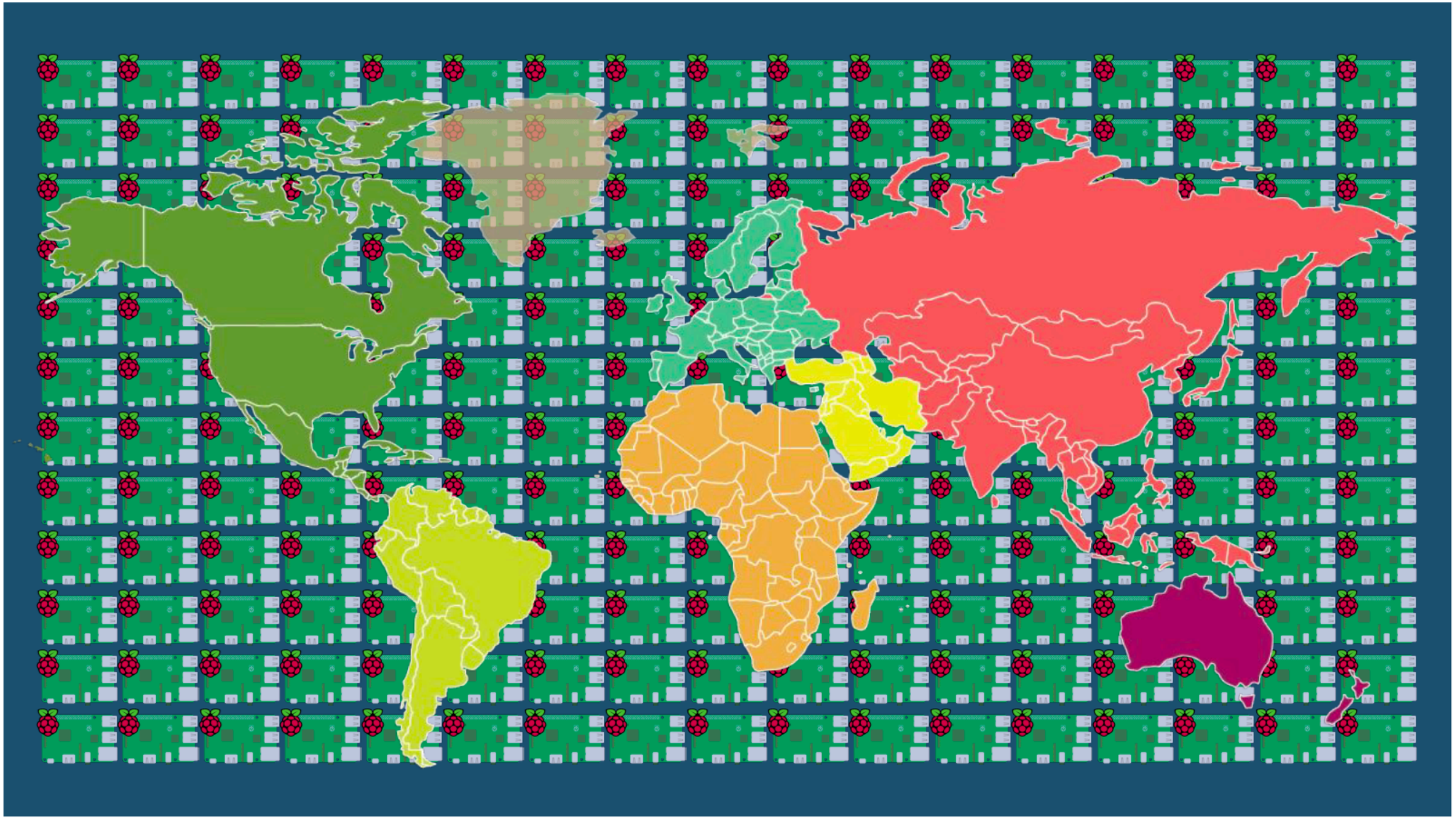
BalenaOS

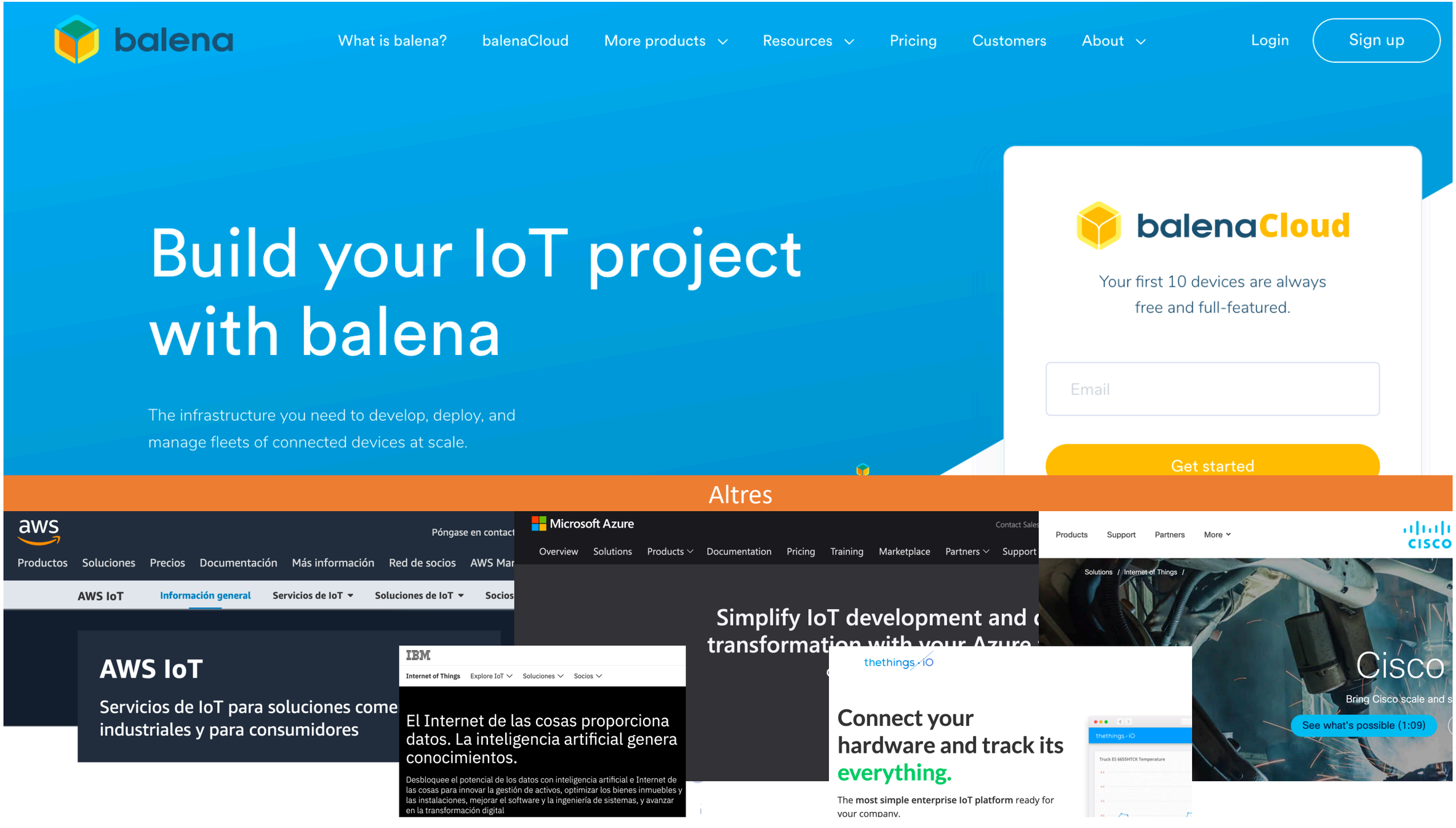

balenaCloud
openbalena
balenaOS
balenaEngine
balenaEtcher
balenaFin





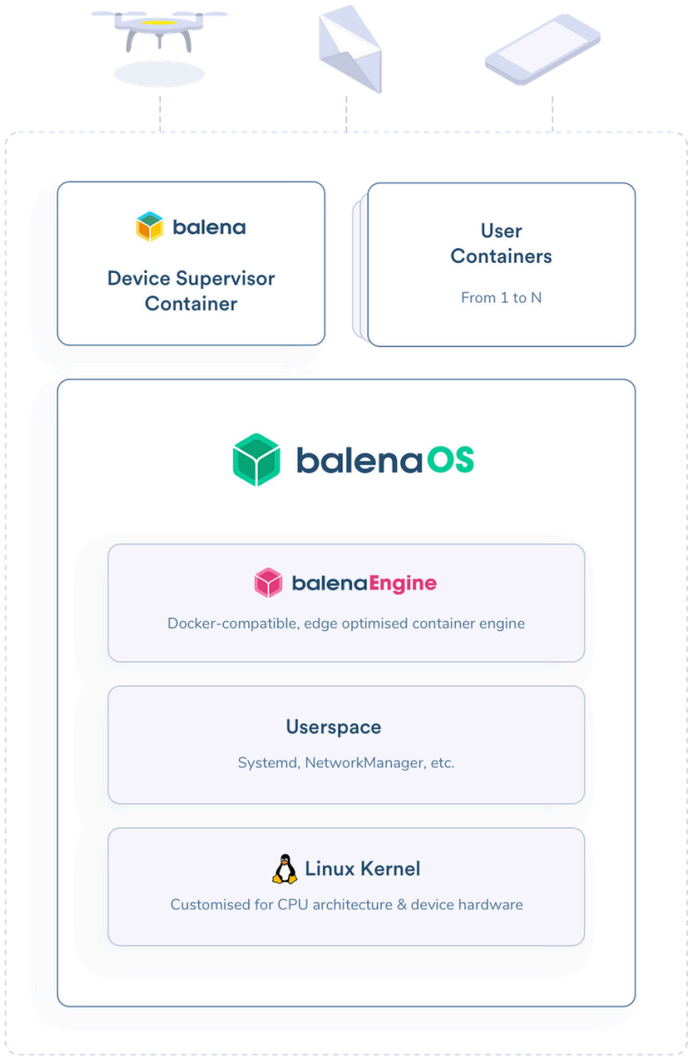


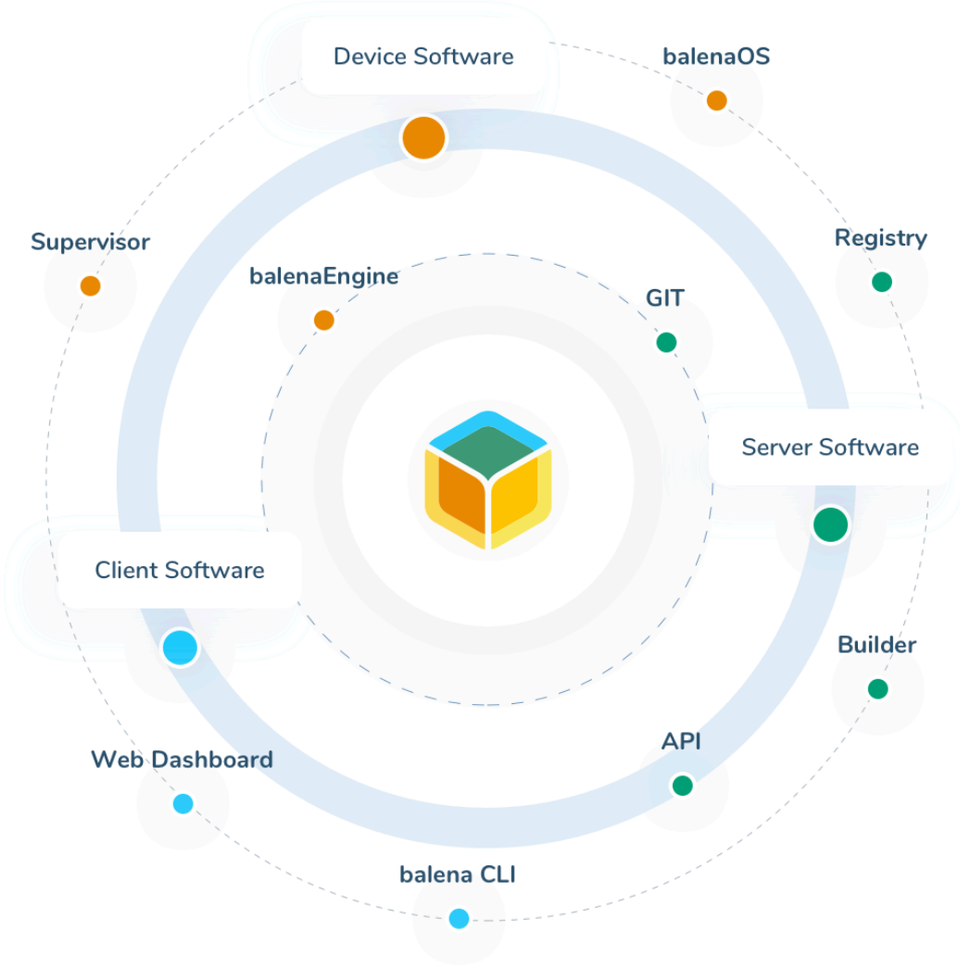



balena
What is balena?
balenaCloud
More products
Resources
Pricing
Customers
About
Login
Sign up
Build your IoT project with balena
The infrastructure you need to develop, deploy, and manage fleets of connected devices at scale.
balenaCloud
Your first 10 devices are always free and full-featured.
Email
Get started
Altres
aws
Productos
Soluciones
Precios
Documentación
Más información
Red de socios
AWS IoT
Información general
Servicios de IoT
Soluciones de IoT
AWS IoT
Servicios de IoT para soluciones comer
industriales y para consumidores
Microsoft Azure
Overview
Solutions
Products
Documentation
Pricing
Training
Marketplace
Partners
Support
Simplify IoT development and
transformation with your Azure
IBM
Internet of Things
Explore IoT
Soluciones
Socios
El Internet de las cosas proporciona datos. La inteligencia artificial genera conocimientos.
Desbloquee el potencial de los datos con inteligencia artificial e Internet de las cosas para innovar la gestión de activos, optimizar los bienes inmuebles y las instalaciones, mejorar el software y la ingeniería de sistemas, y avanzar en la transformación digital
thethings.iO
Connect your hardware and track its everything.
The most simple enterprise IoT platform ready for your company
Products
Support
Partners
More
CISCO
Solutions / Internet of Things /
Cisco
Bring Cisco scale and s
See what's possible (1:09)

BalenaOS

https://www.balena.io/cloud/

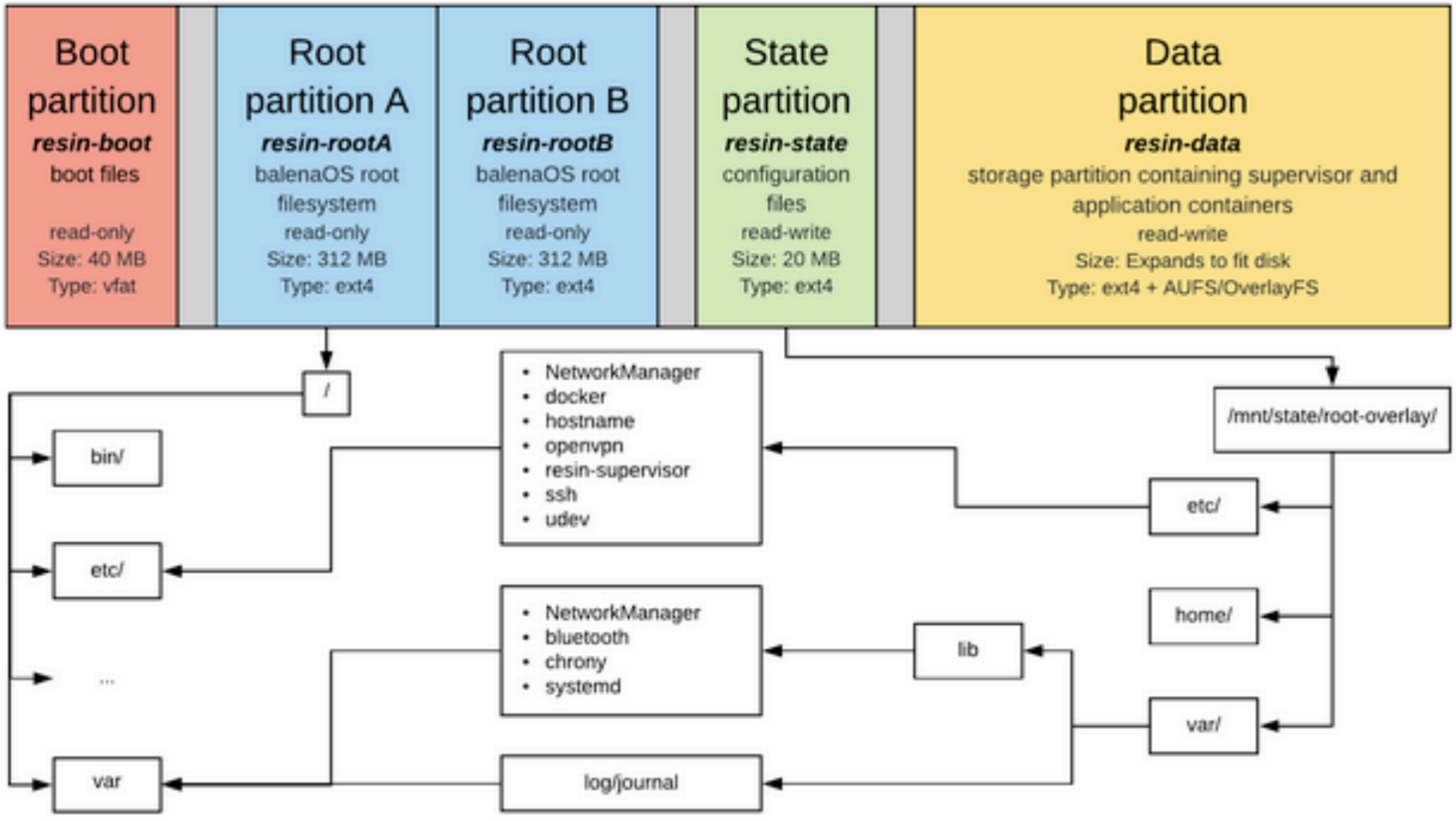




https://www.balena.io/os/docs/architecture/

BalenaOS

https://www.balena.io/cloud/



https://www.balena.io/os/docs/architecture/

BalenaOS

https://www.balena.io/cloud/

Development vs. Production images

Each version of balenaOS is available in development and production variants, both built from the same source, but with slightly differing feature sets. The development images enable a number of useful features while developing, namely:

- Passwordless SSH access into balenaOS on port 22222 as the root user.
- Docker socket exposed on port `2375`, which allows `balena push` / `build` / `deploy`, that enables remote Docker builds on the target device (see Deploy to your Fleet).
- Getty console attached to tty1 and serial.
- Capable of entering local mode for rapid development of application containers locally.

Note: Raspberry Pi devices don't have Getty attached to serial.

Production images disable passwordless root access, and an SSH key must be added to `config.json` to access a production image.

In both development and production versions of balenaOS, logs are written to an 8 MB journald RAM buffer in order to avoid wear on the flash storage used by most of the supported boards.

To persist logs on the device, enable persistent logging by setting the `"persistentLogging": true` key in `config.json`. The logs can be accessed via the host OS at `/var/log/journal`. For versions of balenaOS < 2.45.0, persistent logs are limited to 8 MB and stored in the state partition of the device. balenaOS versions >= 2.45.0 store a maximum of 32 MB of persistent logs in the data partition of the device.

https://www.balena.io/docs/learn/deploy/deployment/

https://www.balena.io/os/docs/architecture/

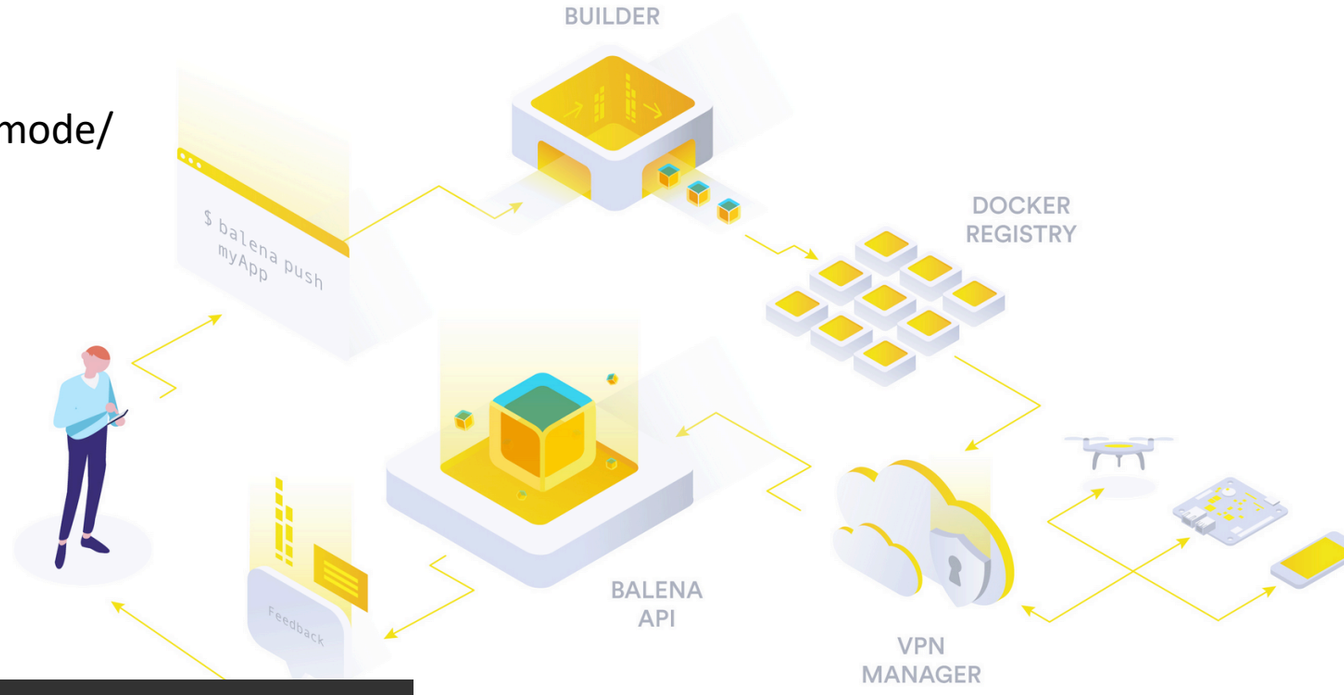
BalenaOS

https://www.balena.io/cloud/

Development

https://www.balena.io/docs/learn/develop/local-mode/

```
FROM balenalib/%%BALENA_MACHINE_NAME%%-node

COPY package.json /package.json
RUN npm install

COPY src/ /usr/src/app
CMD ["node", "/usr/src/app/main.js"]
```



Deployment

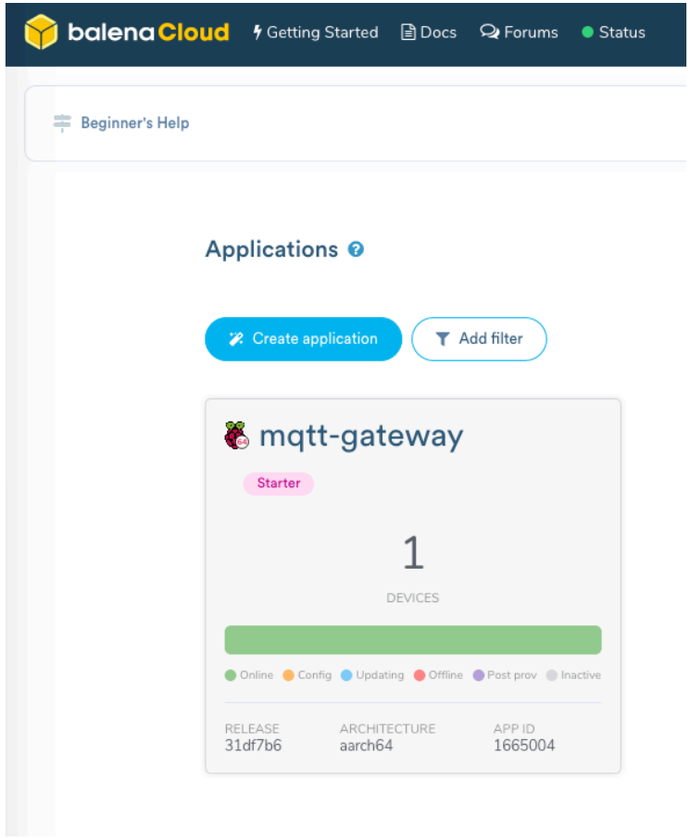
```
$ balena push myApp --emulated
[Info]     Starting build for myApp, user balena_projects
[Info]     Dashboard link: https://dashboard.balena-cloud.com/apps/1426783/devices
[Info]     Running locally emulated build
[Info]     Pulling previous images for caching purposes...
[Success]  Successfully pulled cache images
[main]     Step 1/2 : FROM balena/secret_sauce
[main]      ---> 6e48e49f10a6
[main]     Step 2/2 : CMD cat /etc/os-release
```

BalenaOS

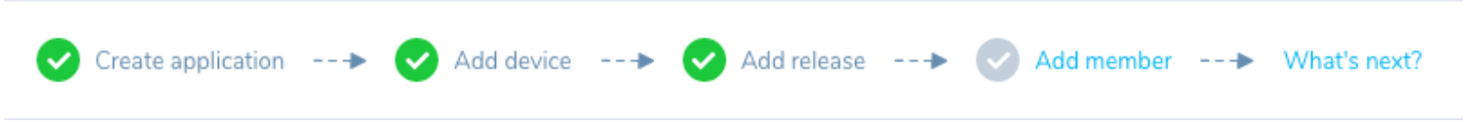
https://www.balena.io/cloud/

Projecte: Servei MQTT connectat amb una base de dades Mongo, per historificar tots els missatges enviats per MQTT

1) Crear una compta a BalenaCoud

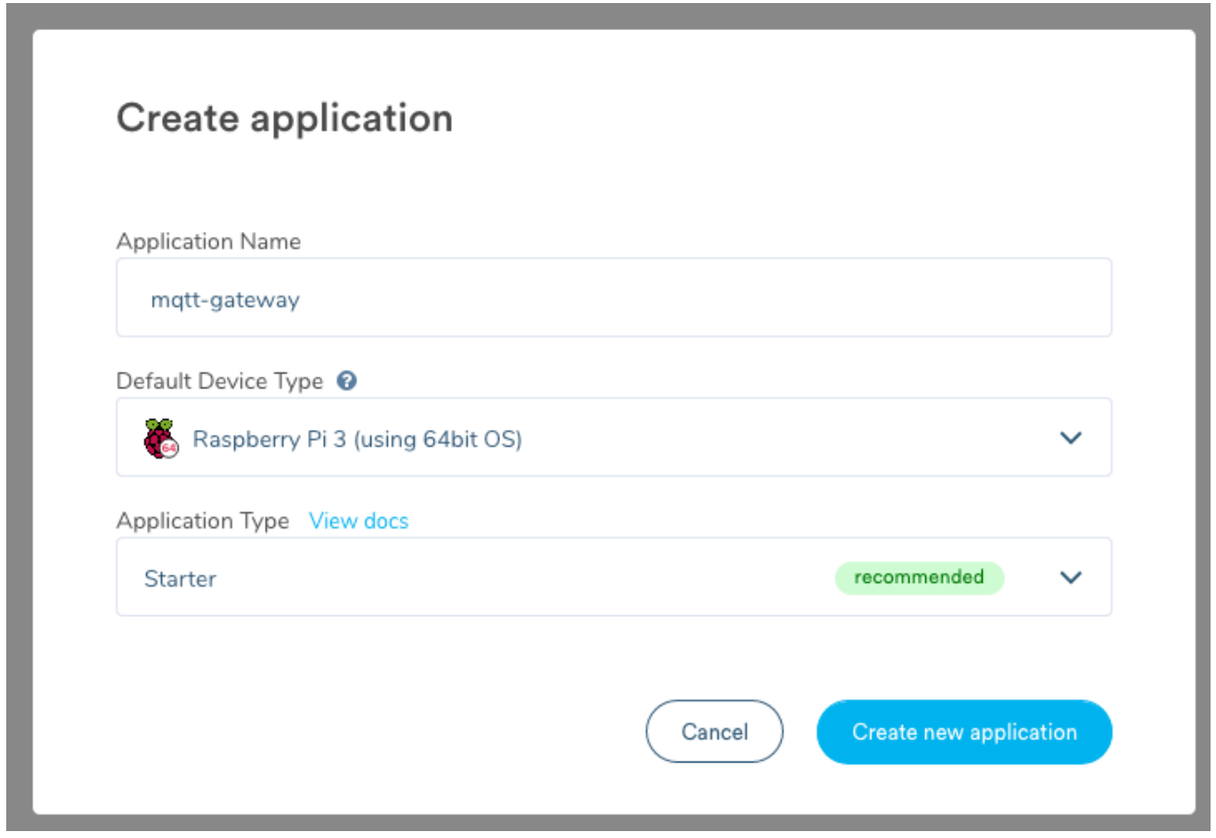


2) Seguirem les passes...



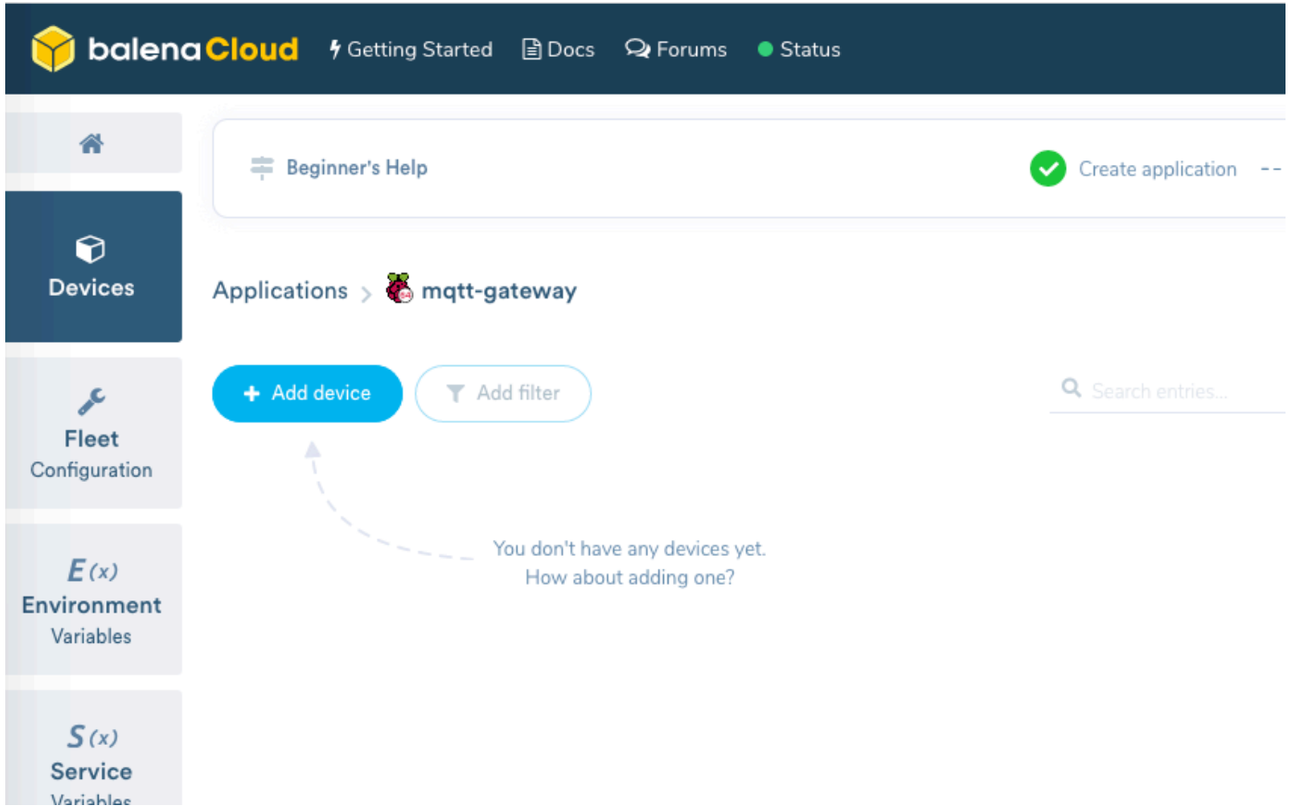
Setup BalenaOS :: Crear la Aplicació

https://www.balena.io/docs/learn/getting-started/raspberrypi3/python/



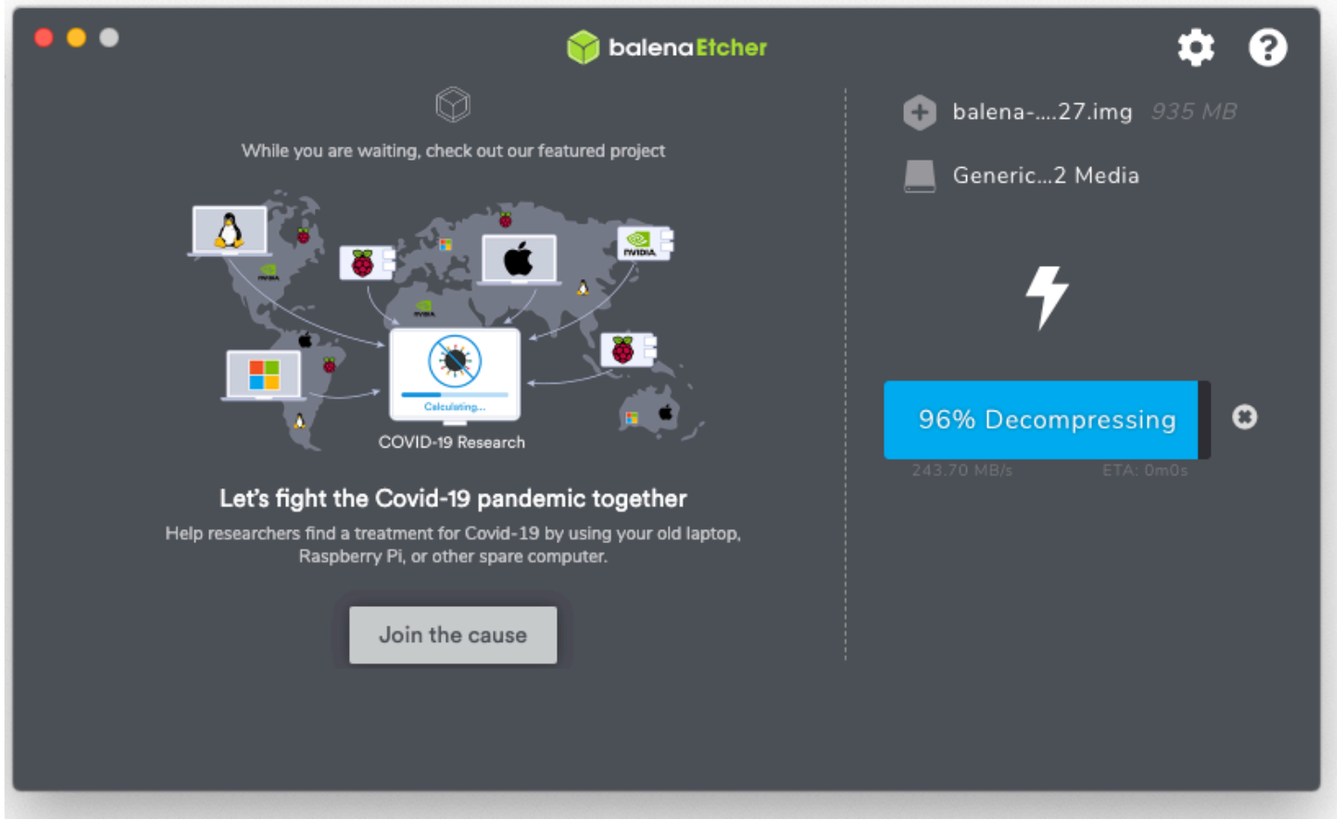
Setup BalenaOS :: Afegir dispositius

https://www.balena.io/docs/learn/getting-started/raspberrypi3/python/



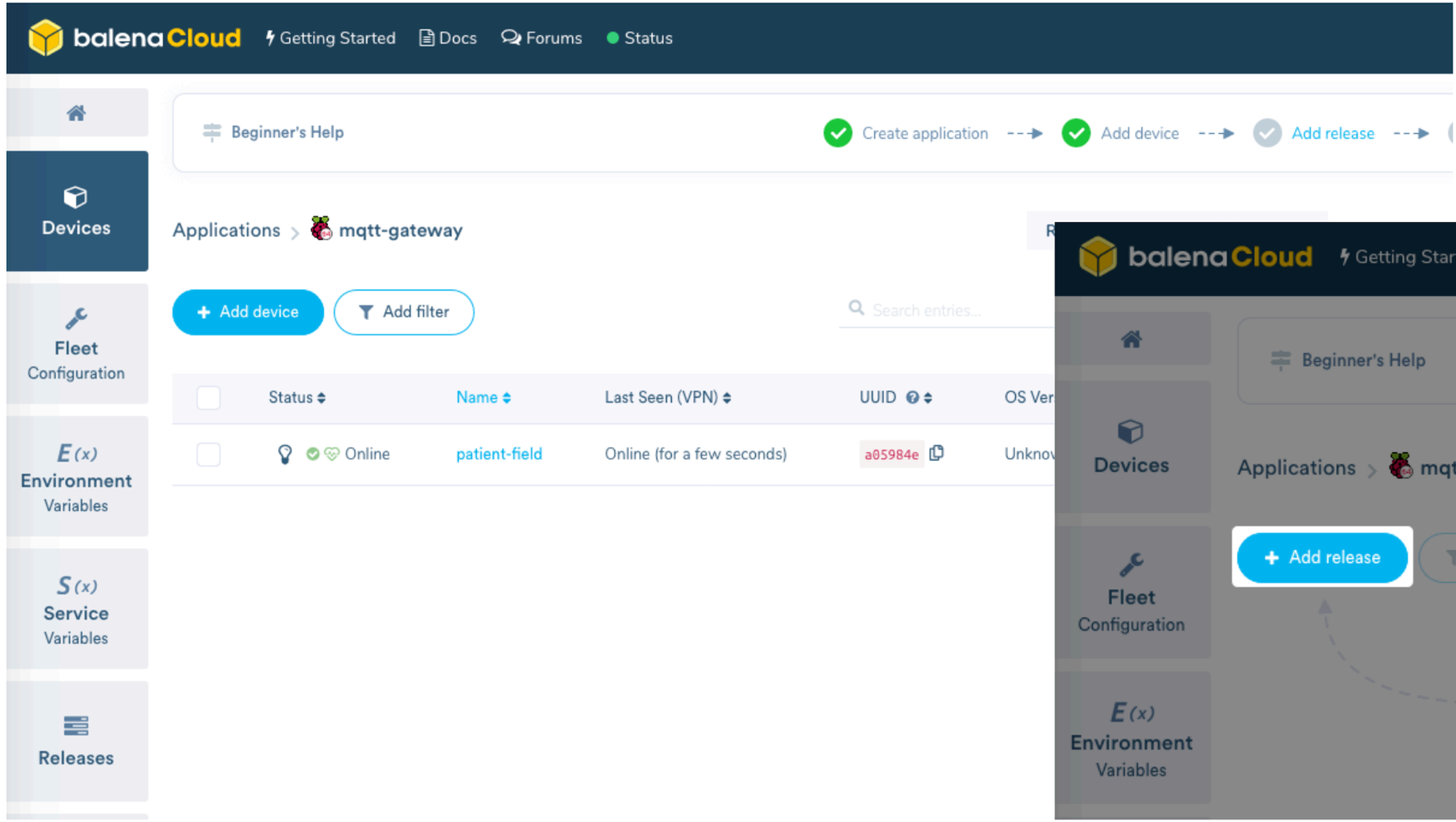
Setup BalenaOS :: Cremar la SD

https://www.balena.io/docs/learn/getting-started/raspberrypi3/python/



Setup BalenaOS :: Afegir release de codi

https://www.balena.io/docs/learn/getting-started/raspberrypi3/python/



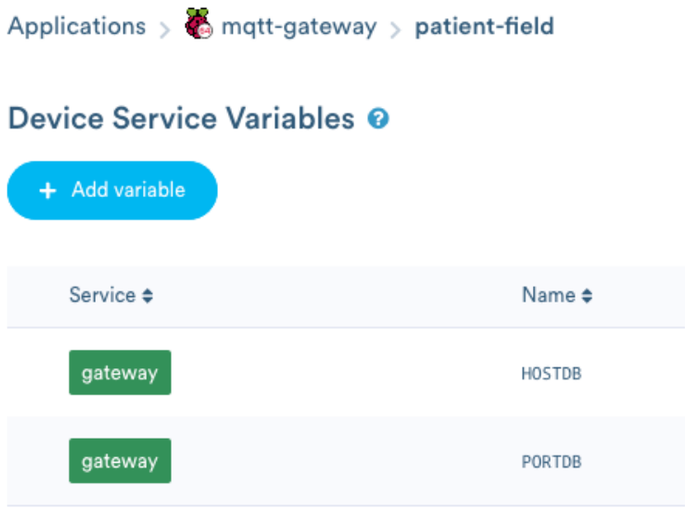
Setup BalenaOS :: Git clone del projecte

https://github.com/balena-io/balena-cli/blob/master/INSTALL.md

git clone https://github.com/davidraba/edu-masteriot-fleet-management.git

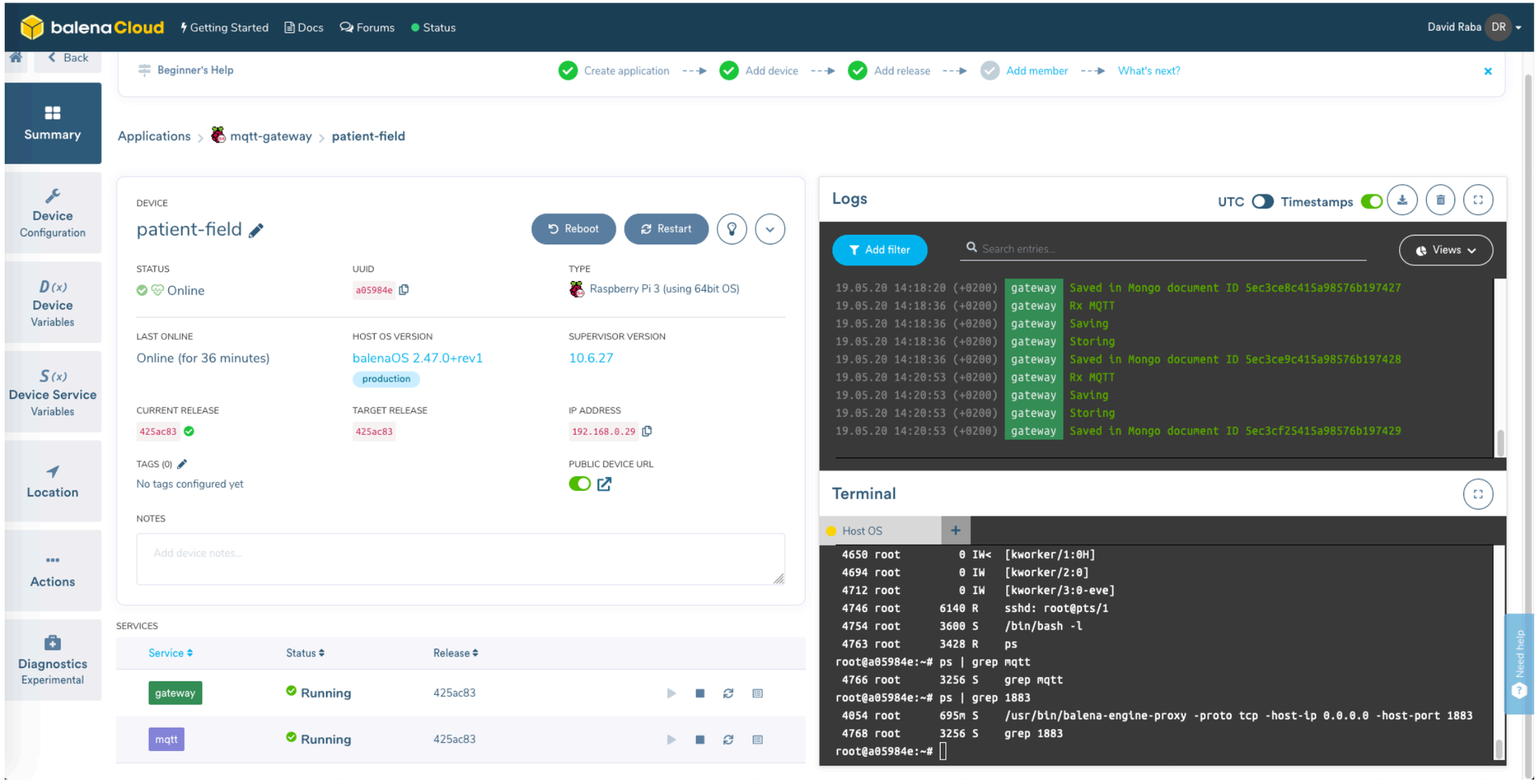
From bash:

$ balena login
$ balena push mqtt-gateway --source ./



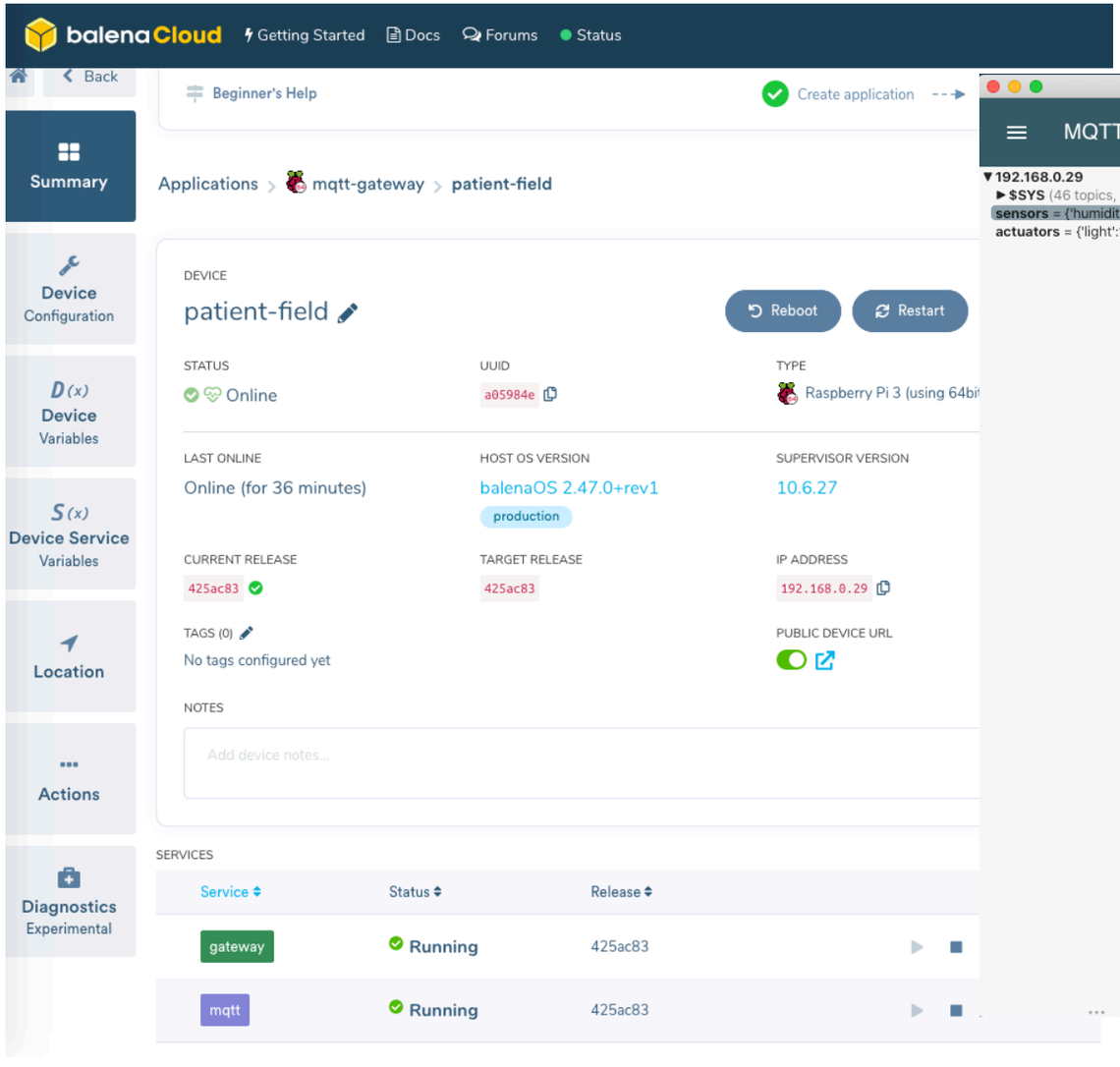

```
 _                 _
| |__   __ _ | | ___ _ __   __ _
| '_ \ / _` || |/ _ \ '_ \ / _` |
| |_) | (_) || ||  __/ | | | (_) |
|_.__/ \__,_||_| \___|_| |_| \__,_|


Logging in to balena-cloud.com
? How would you like to login? Web authorization (recommended)
Connecting to the web dashboard
Successfully logged in as: gh_davidraba

Find out about the available commands by running:

  $ balena help

If you need help, or just want to say hi, don't hesitate in reaching out
through our discussion and support forums at https://forums.balena.io

For bug reports or feature requests, have a look at the GitHub issues or
create a new one at: https://github.com/balena-io/balena-cli/issues/
[Info]       Starting build for mqtt-gateway, user gh_davidraba
[Info]       Dashboard link: https://dashboard.balena-cloud.com/apps/1665004/devices
[Info]       Building on arm01
[Info]       Pulling previous images for caching purposes...
[Success]    Successfully pulled cache images
```

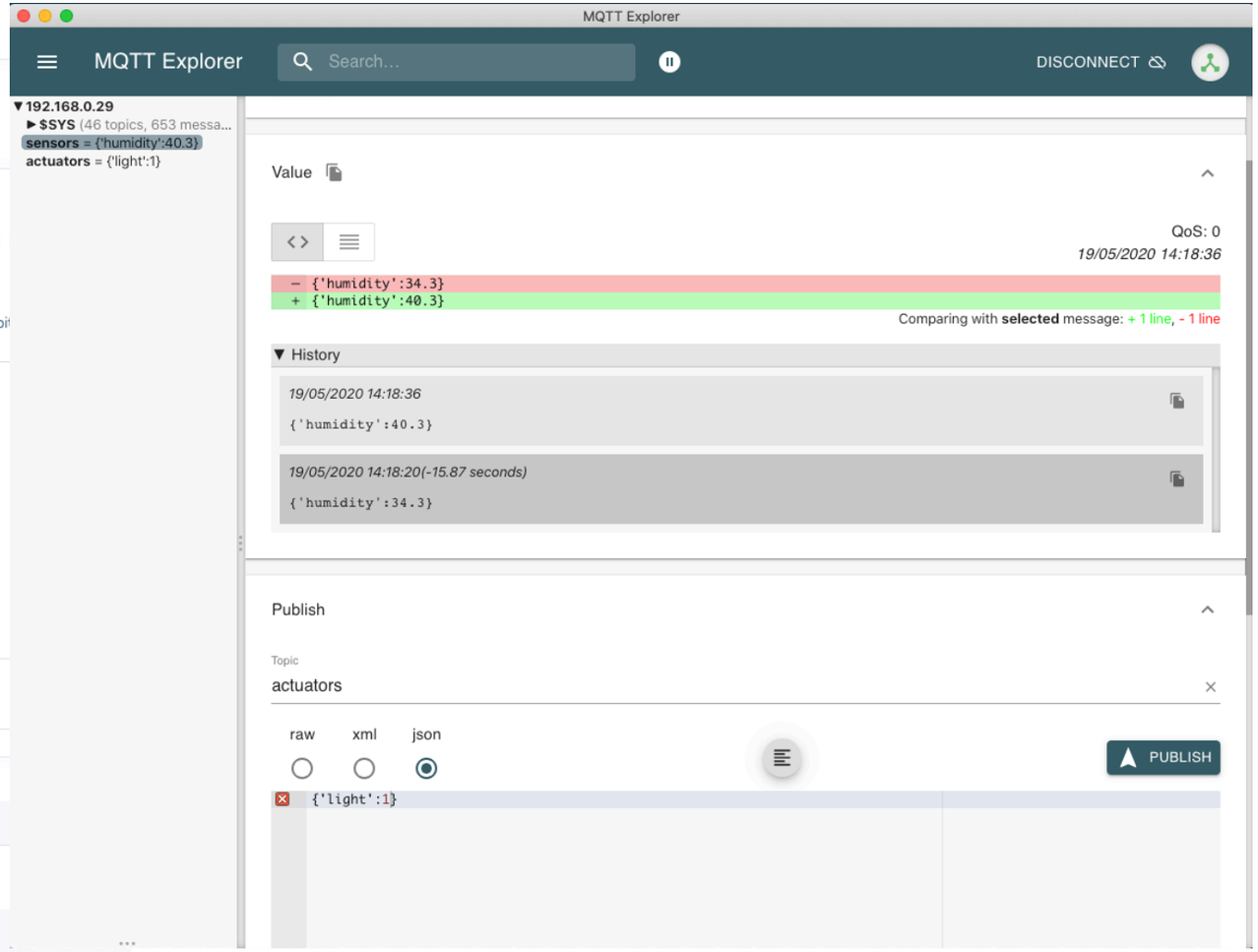
Setup BalenaOS :: up-and-running



Setup BalenaOS :: Simularem un missatge MQTT

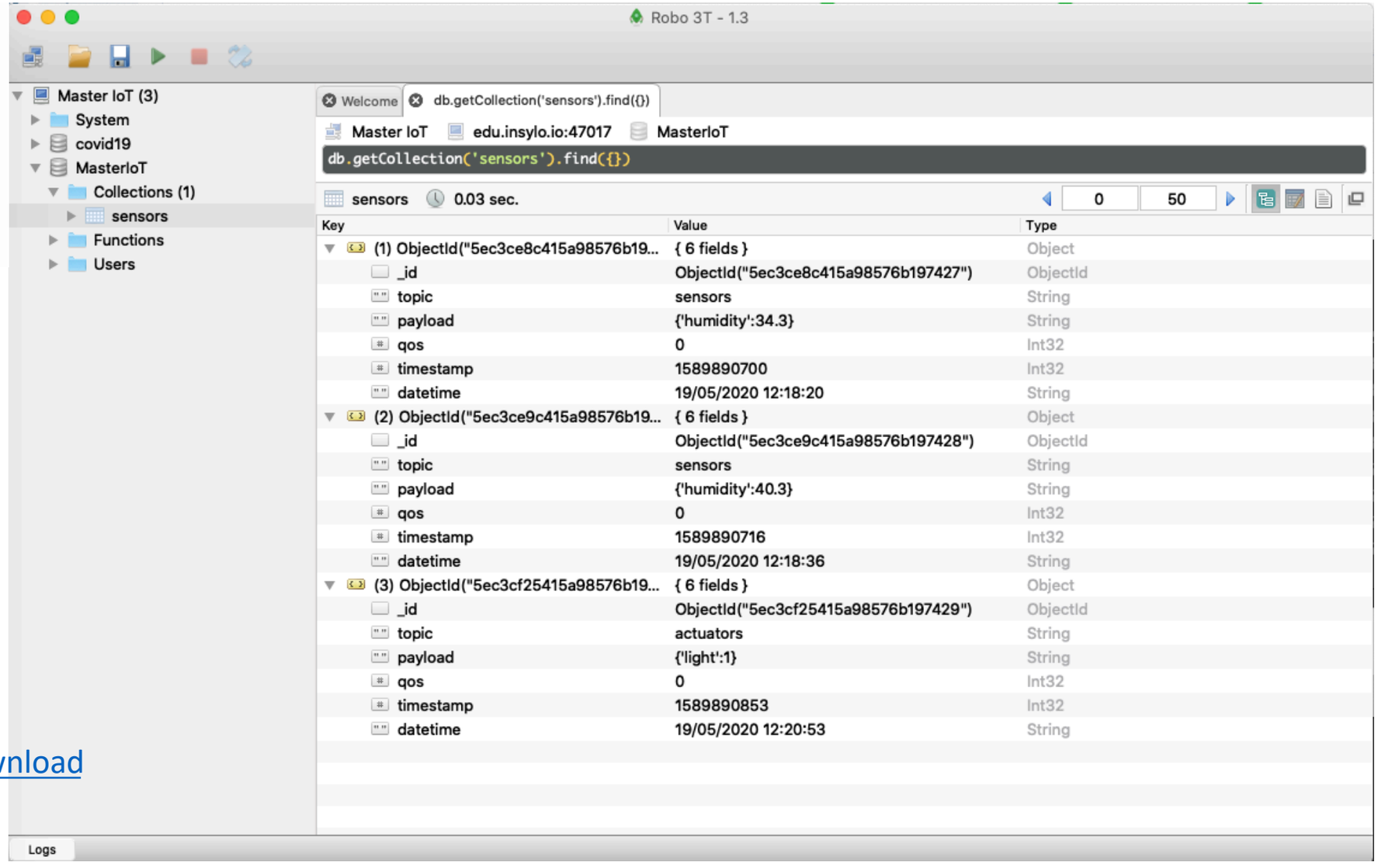
http://mqtt-explorer.com/





Setup BalenaOS :: Revisem la BBDD

Robo 3T



https://robomongo.org/download

BalenaOS :: Advanced stuff...

Local Mode

Local mode allows you to build and sync code to your device locally for rapid development.

Multicontainers

Develop an application with multiple containers to provide a more modular approach to application management.

Configurations

Manage your device fleet with the use of configuration, environment, and service variables

Get Inspired

Explore our projects to give you an idea of more things you can do with balena.

How to get help?

If you find yourself stuck or confused, help is just a click away.

GPIO

```
gpio:
    build: ./gpio
    devices:
      - "/dev/i2c-1:/dev/i2c-1"
      - "/dev/mem:/dev/mem"
      - "/dev/ttyACM0:/dev/ttyACM0"
    cap_add:
      - SYS_RAWIO
```

https://www.balena.io/docs/learn/develop/hardware/i2c-and-spi/

https://www.balena.io/docs/learn/manage/configuration/

https://www.balena.io/docs/learn/develop/multicontainer/

https://hackmd.io/@garethtdavies/BJNh35uiH

Referencies – BalenaOS / Docker

- How to burn image with python based environment
  - https://www.balena.io/docs/learn/getting-started/raspberrypi3/python/
- Hello world Python service
  - https://github.com/balena-io-examples/balena-python-hello-world
- Integració amb plataforma de injesta de logs Datahog
  - https://github.com/balena-io-playground/balena-datadog
- Deploy de servei Mongo
  - https://github.com/balena-io-playground/express-mongo-sample
- Definició de serveis via Docker
  - https://www.balena.io/docs/learn/develop/dockerfile/
- Definició i ús del GPIO RaspPi
  - https://github.com/balena-io-playground/balena-rpi-nodejs-basic-gpio
- Aplicació GPS
  - https://scrobotics.es/2020/02/03/gps-tracker-with-balenaos/
- Arquitectura: https://www.balena.io/os/docs/architecture/