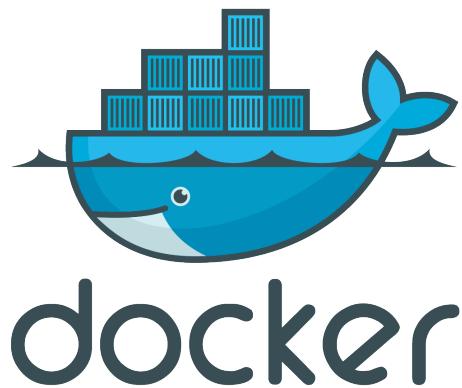


1



2



# Bases de dades i Gestió de flotes de dispositius

**Màster Internet of Things**

25 i 27 de 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](http://www.fliwer.com) and [www.fliwerpro.com](http://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 from online traffic collection.
- Core development of an online benchmarking system based on traffic sniffing
- Startup experience
- Ruby on Rails | Ruby | HTML | CSS | C++ | Pound | Apache



# Start-ups

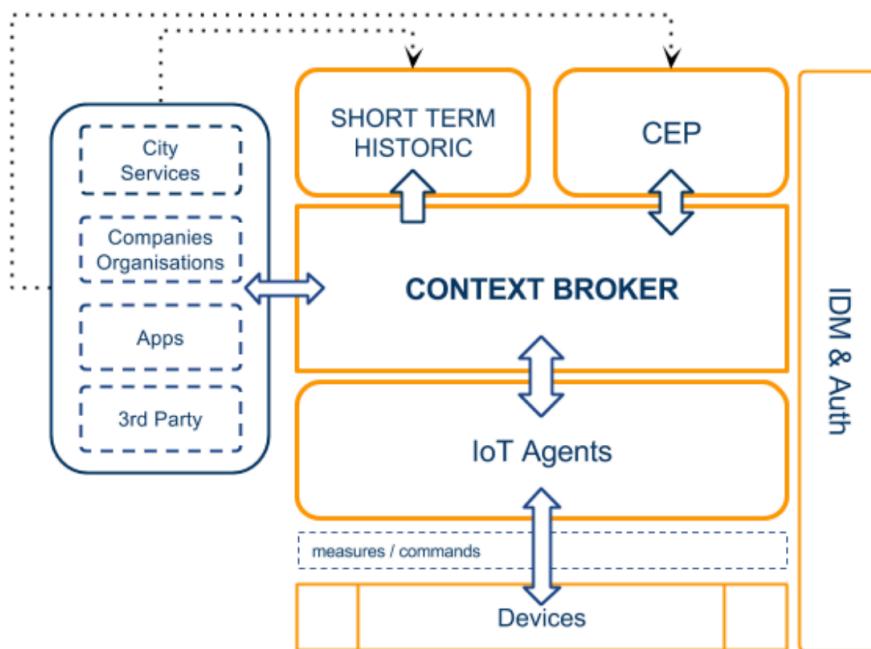


## Development & Project Manager - Sporadically CTO

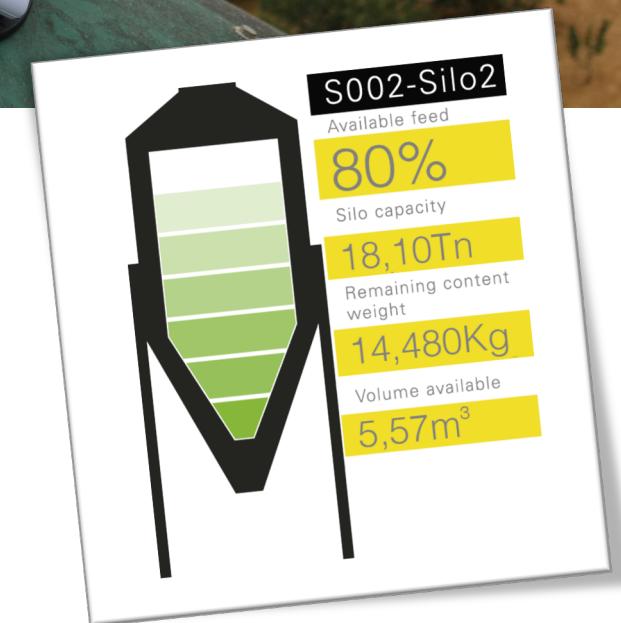
INSYLO TECHNOLOGIES SL

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.



25.5.2020



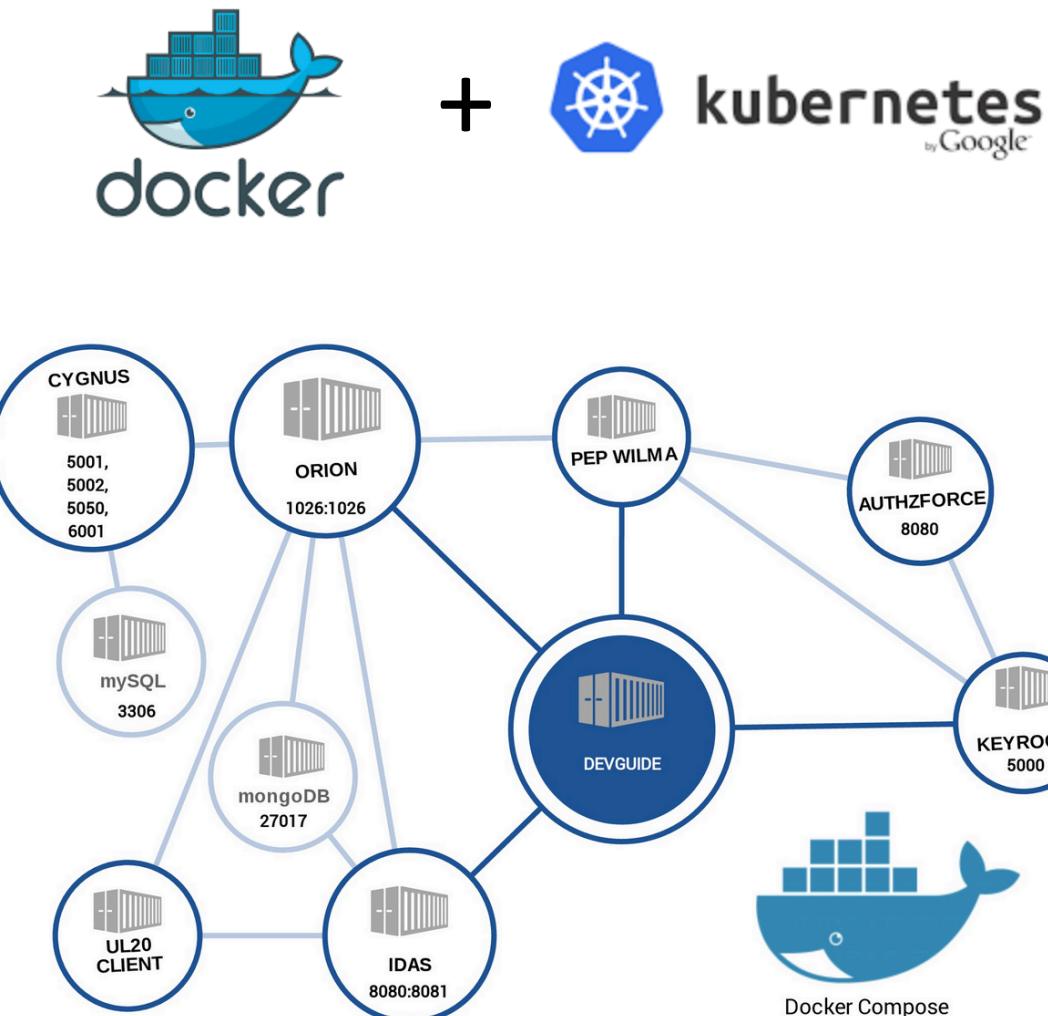
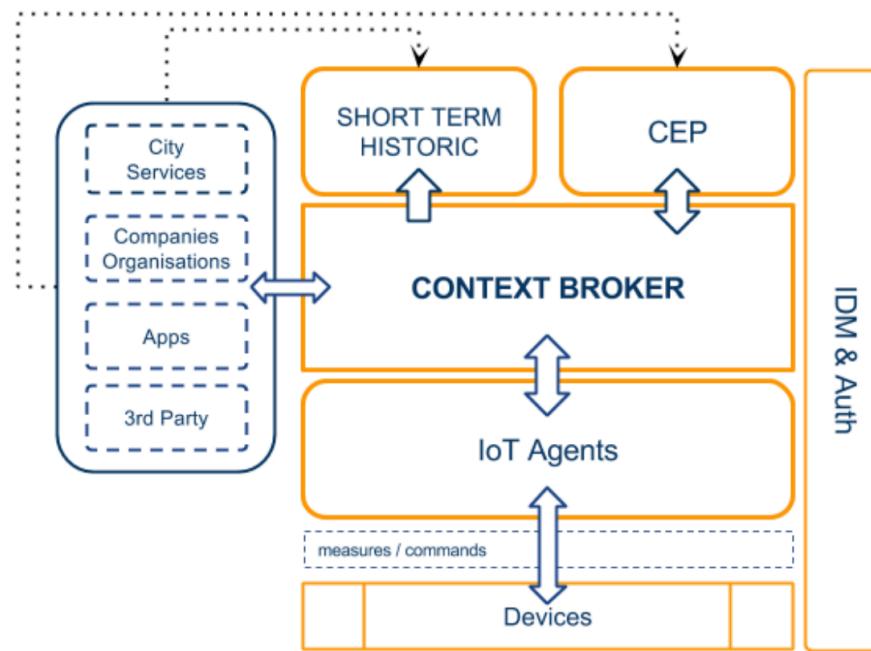
# Start-ups



## 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.



- **Sessió 1**

- Docker
- BalenaOS
- Definició de serveis a través de Docker
- Deploy de serveis a través de BalenaOS

Sessió integralment Online

Activitats prèvies a la sessió:

- 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

- **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 dispositiu de la RaspPi
  - Creació d'un tercer servei que reporti informació relevant via el servei local MQTT **(A casa)**
  - Realització dels consultes per explorer la BBDD generada. **(A casa)**

# 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 eren carregats i descarregats a mà 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 seu principi deia que “*Un vaixell només guanya diners quan està navegant*”

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

A més, de dotar de fiabilitat al sistema.



[https://en.wikipedia.org/wiki/Malcom\\_McLean](https://en.wikipedia.org/wiki/Malcom_McLean)

# Què son els containers?



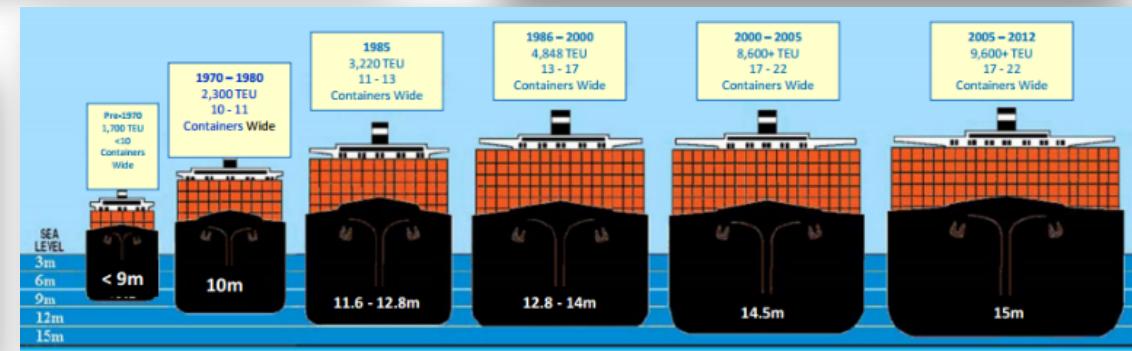
- Omplir el container



# 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 (Imctfy, 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.

## 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



19/9: UNIX v /

## 2001: Linux vServer

Like FreeBSD Jails, [Linux VServer](#) is a jail mechanism that can partition resources (file systems, network addresses, memory) on a computer system. Introduced in 2001, this operating system virtualization that is implemented by patching the Linux kernel. Experimental patches are still available, but the last stable patch was released in 2006



## 2004: Oracle Solaris Containers

2004 Oracle released a [Solaris Container](#) that combines system resource controls and boundary separation provided by zones, which were able to leverage features like snapshots and cloning from ZFS.



## 2005: OpenVZ (Open Virtuzzo)

This is an operating system-level virtualization technology for Linux which uses a patched Linux kernel for virtualization, isolation, resource management and checkpointing. The code was not released as part of the official Linux kernel.



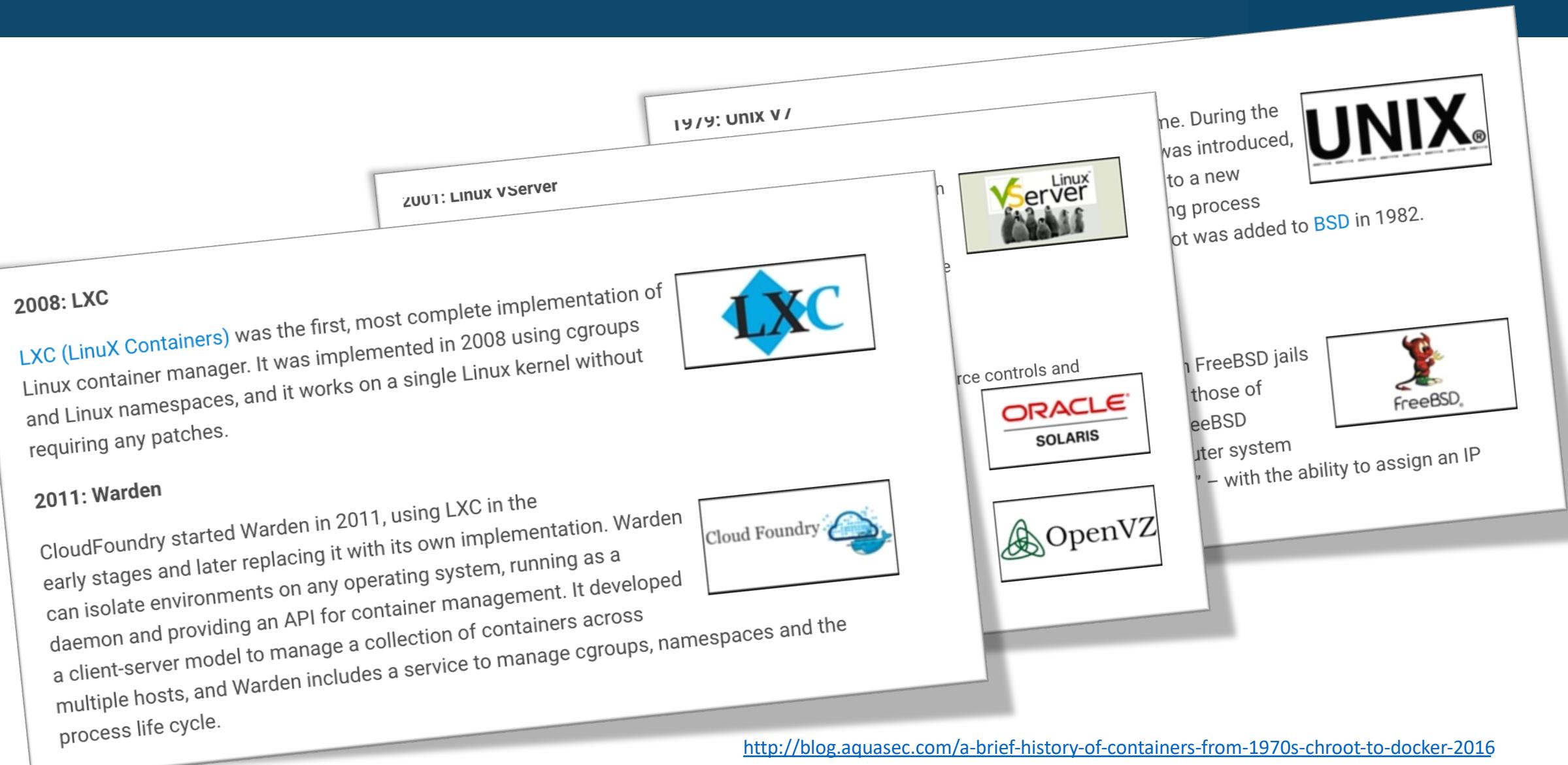
ne. During the was introduced, to a new ing process ot was added to [BSD](#) in 1982.



In FreeBSD jails those of FreeBSD outer system " – with the ability to assign an IP

<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>



## 2013: Docker and the Future

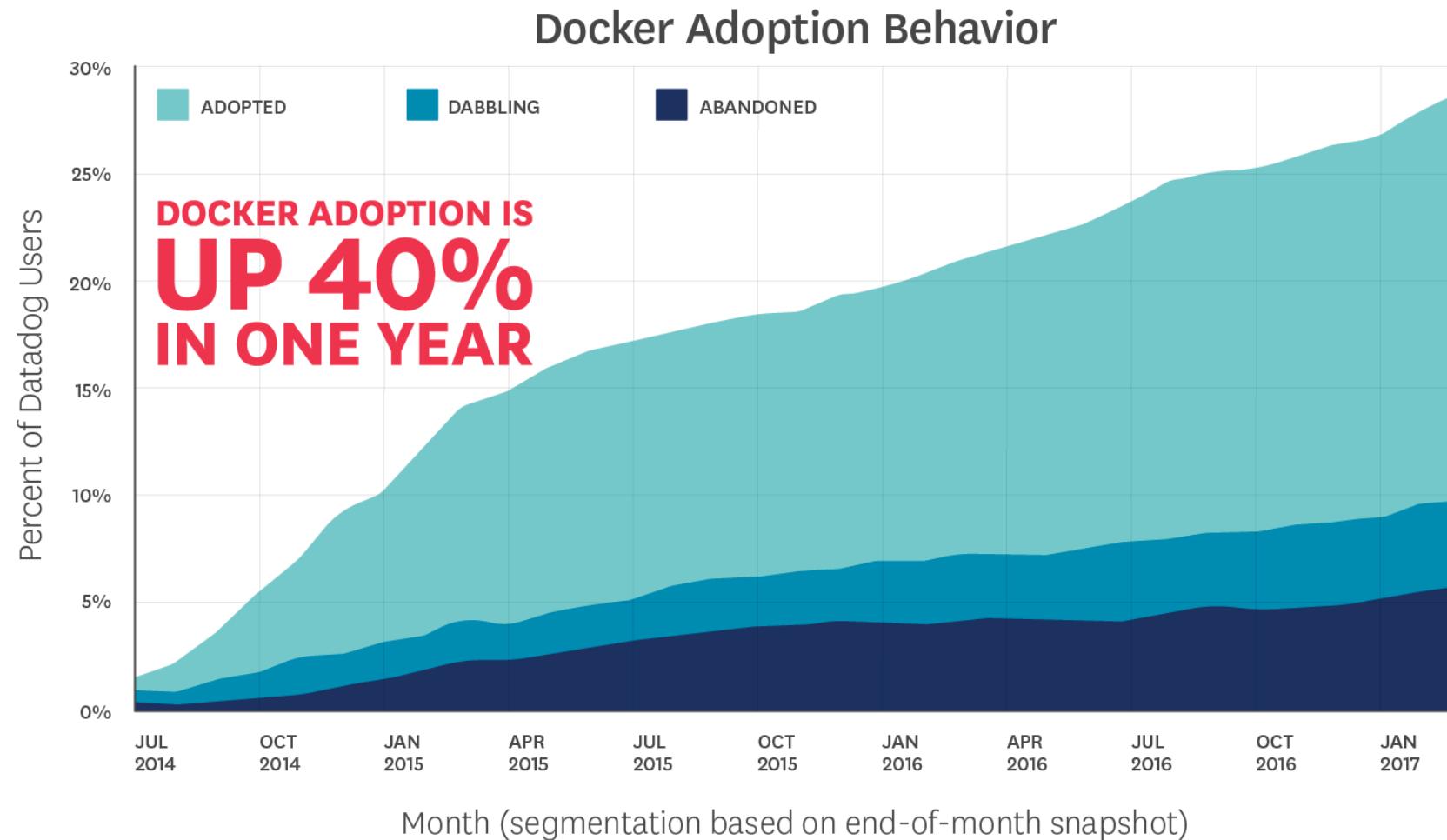
That's my (not so brief) summation of the pre-Docker container landscape. All those iterations had their adopters and devotees, but when Docker emerged in 2013, containers exploded in popularity. It's no coincidence the growth of Docker and container use goes hand-in-hand.

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.

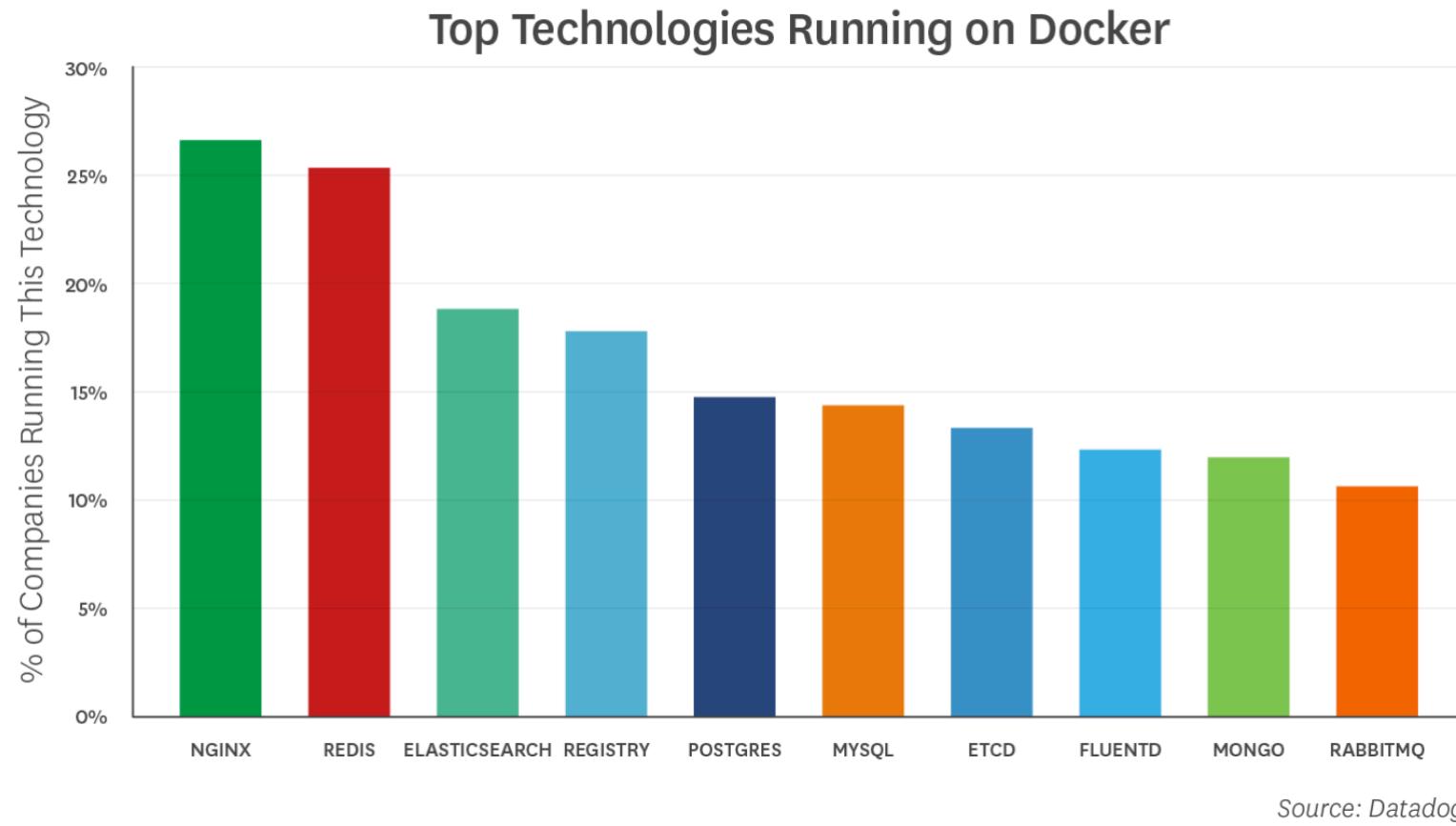
With Docker, developers can create and run application containers quickly. And with the release of Docker Hub, developers can download and run application containers even faster.

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

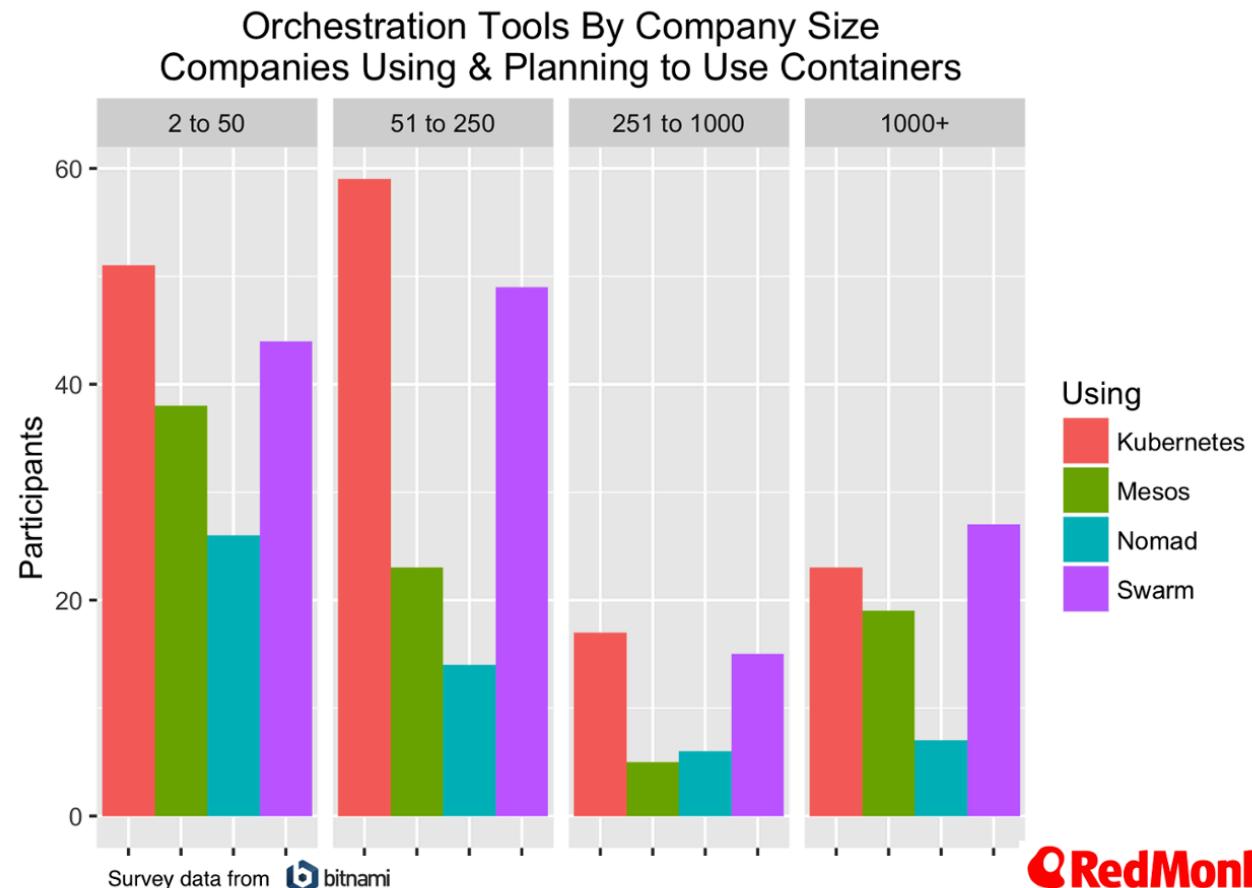
# Utilització de docker



# Utilització de docker



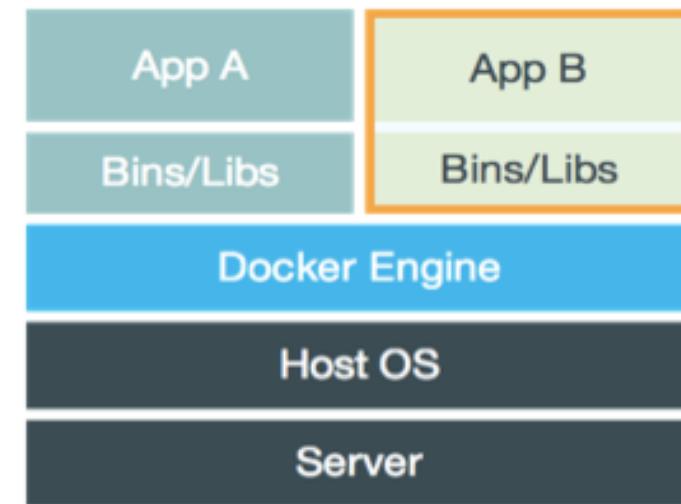
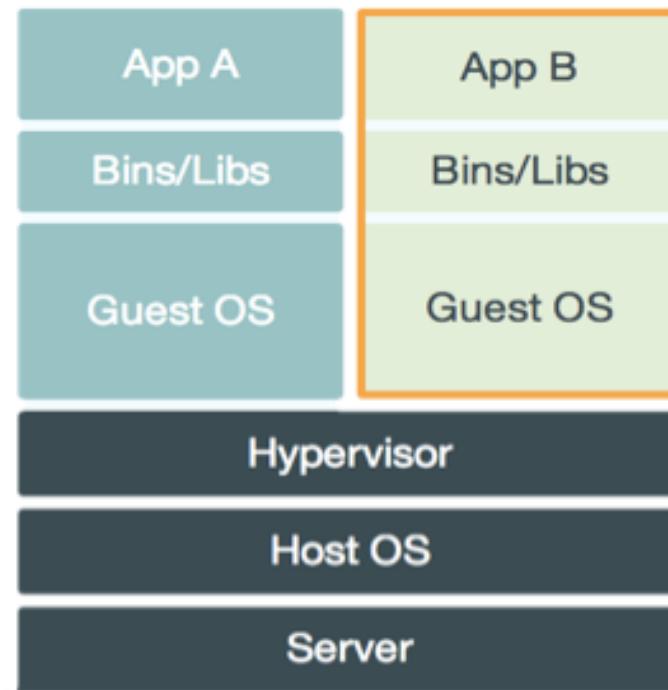
# Utilització de docker



# Docker vs Virtual Machines



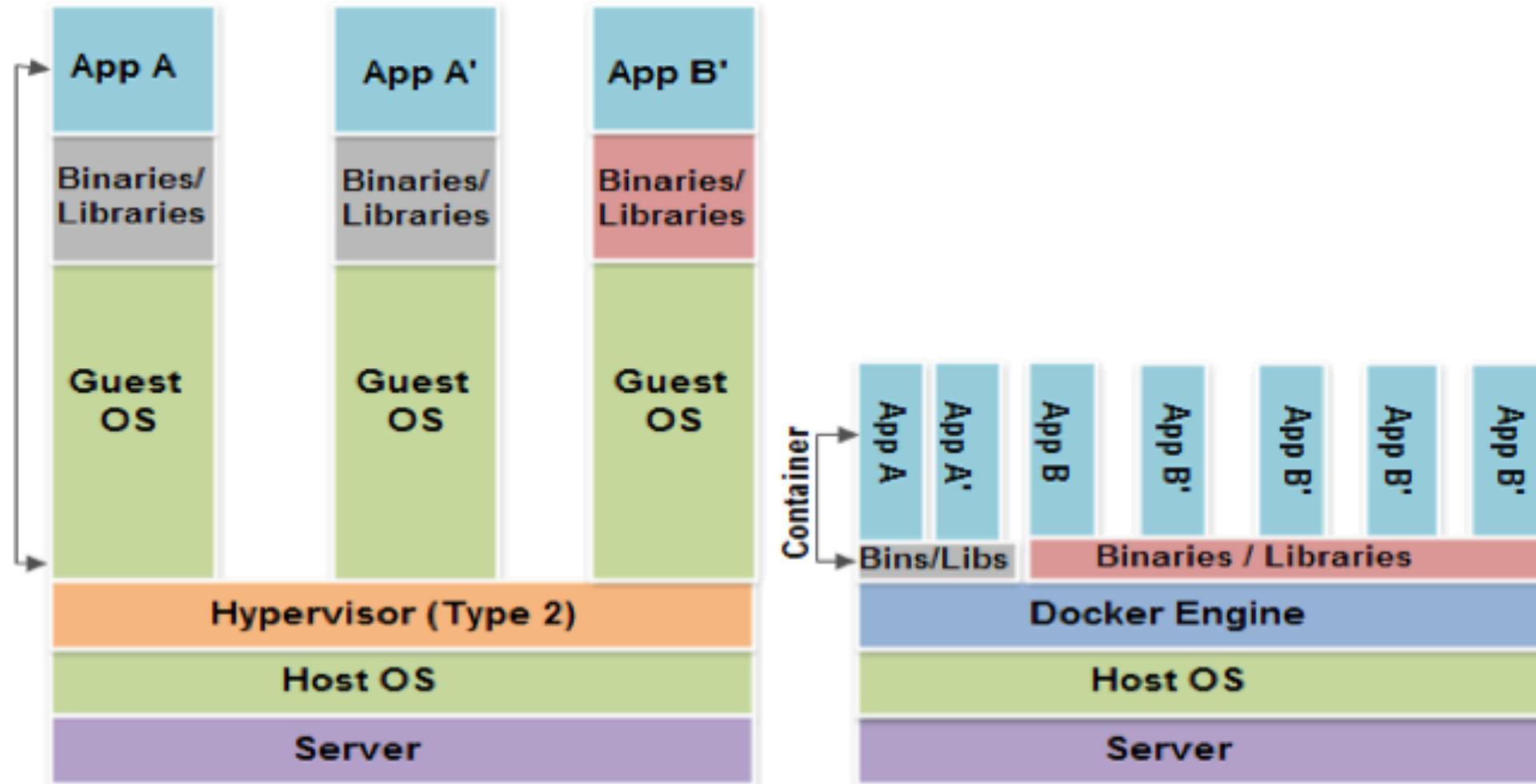
- VM's
  - Instància completa del sistema Operatiu
  - No és fàcil la multi-instància
- Containers
  - Porcions del SO + components
  - Fàcil de duplicar, iniciar i aturar
  - OS Lleuger (Tiny Core Linux & Windows Server Core)



# Docker vs Virtual Machines



## Containers 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



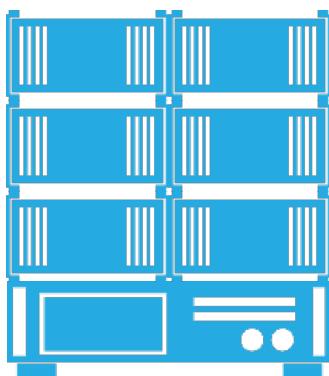
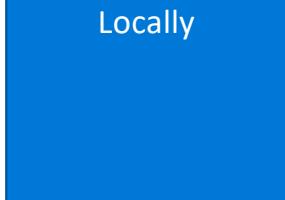
- Requirements mínims:
  - 256-512MB of RAM
  - 1GHz (x86) or 1.4Ghz (x64)CPU
- Recomanats
  - 512MB+
  - 2GHz+ CPU

# On els podem posar?



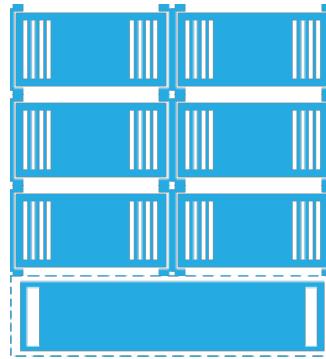
Locally with:

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



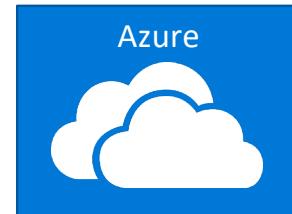
Physical Servers

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



## Clouds

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

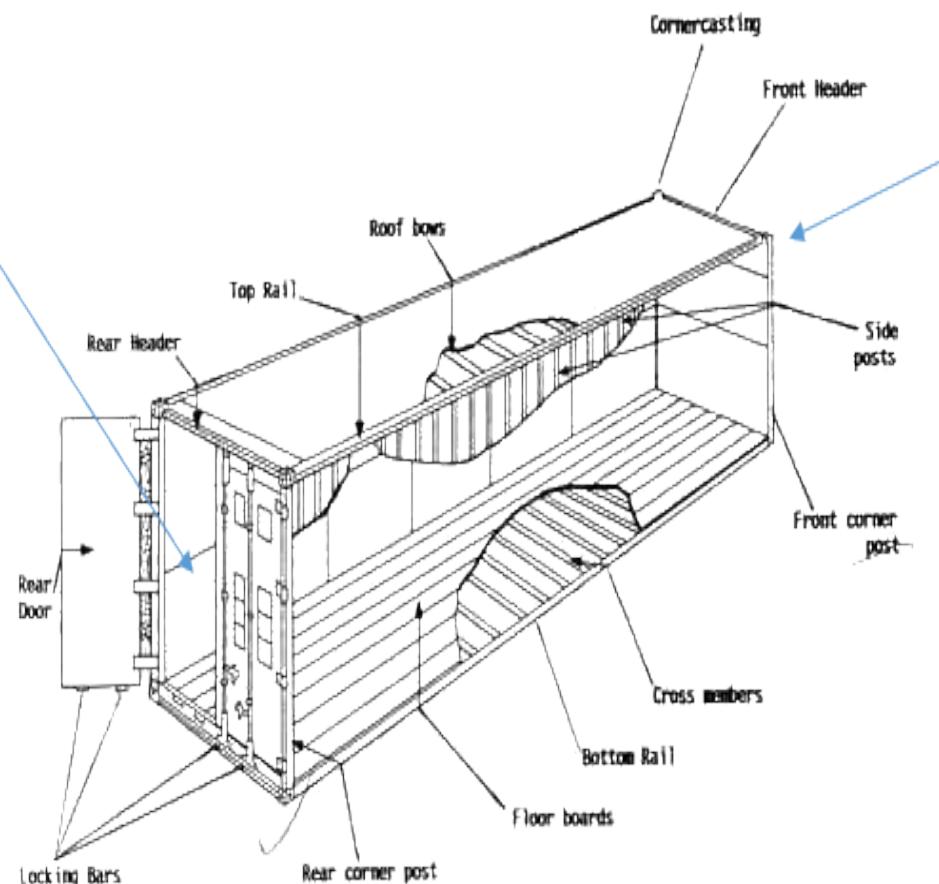


# Què en treiem de bo?



## 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

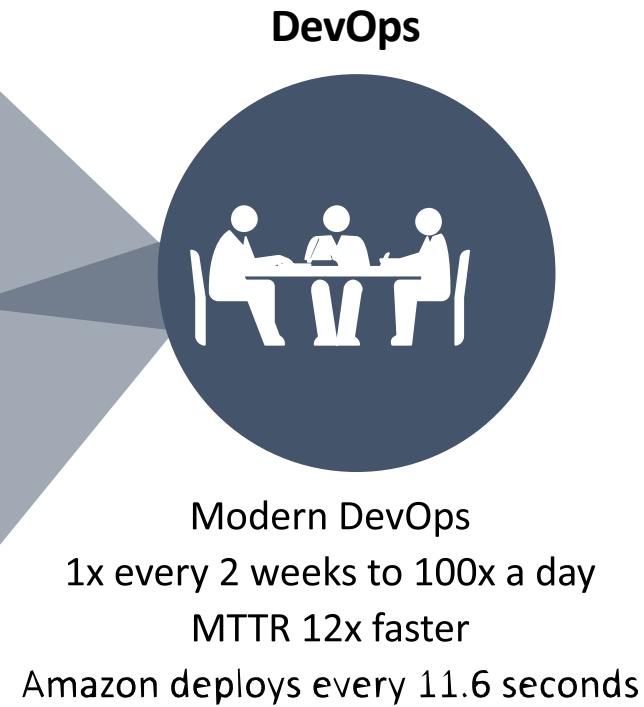


Major components of the container:

## 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?



# Què en treiem de bo?

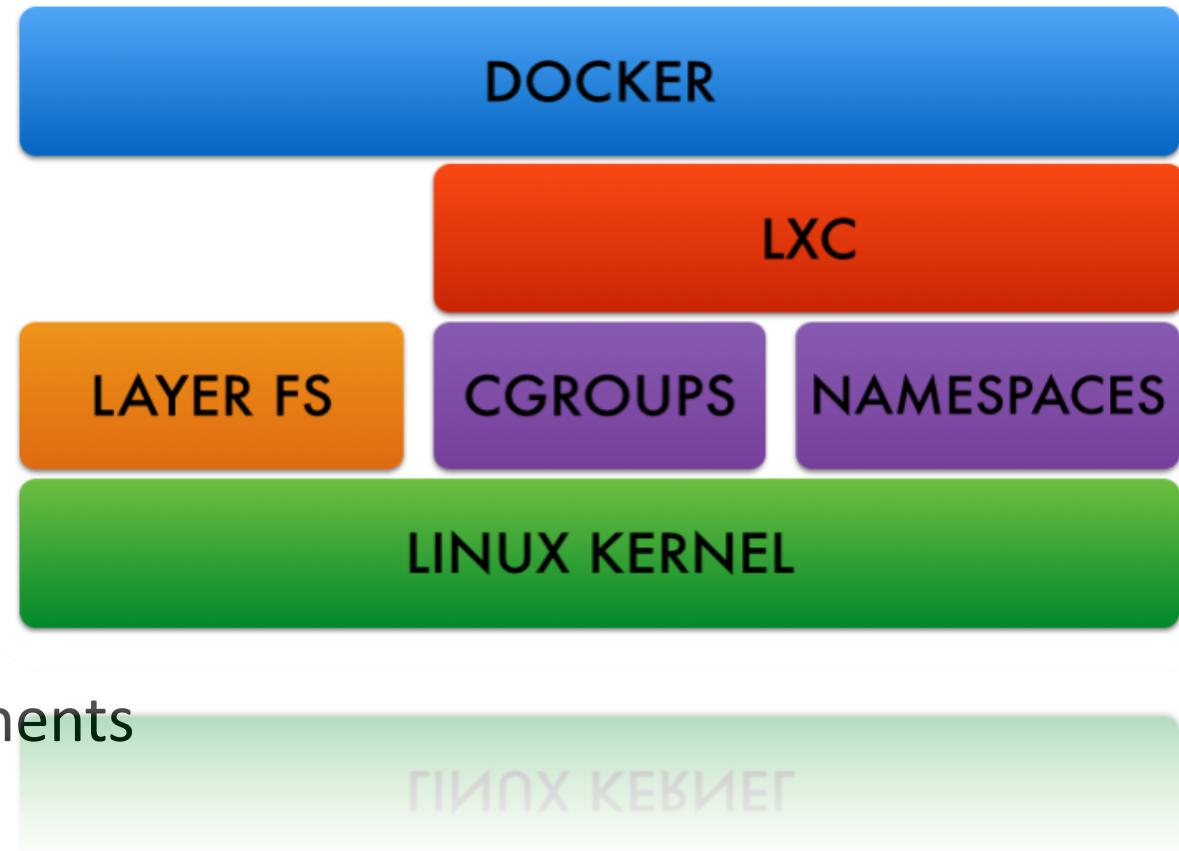


- La infraestructura es converteix en inmutable
- Inici ràpid
- Portable & lleuger
- Tenim una unitat de deploy
- Fàcil creació
- Cada container por ser una porció de la gran aplicació
  - Podem tenir múltiples containers que serveixes a més d'una aplicació (microserveis)

# Terminologia



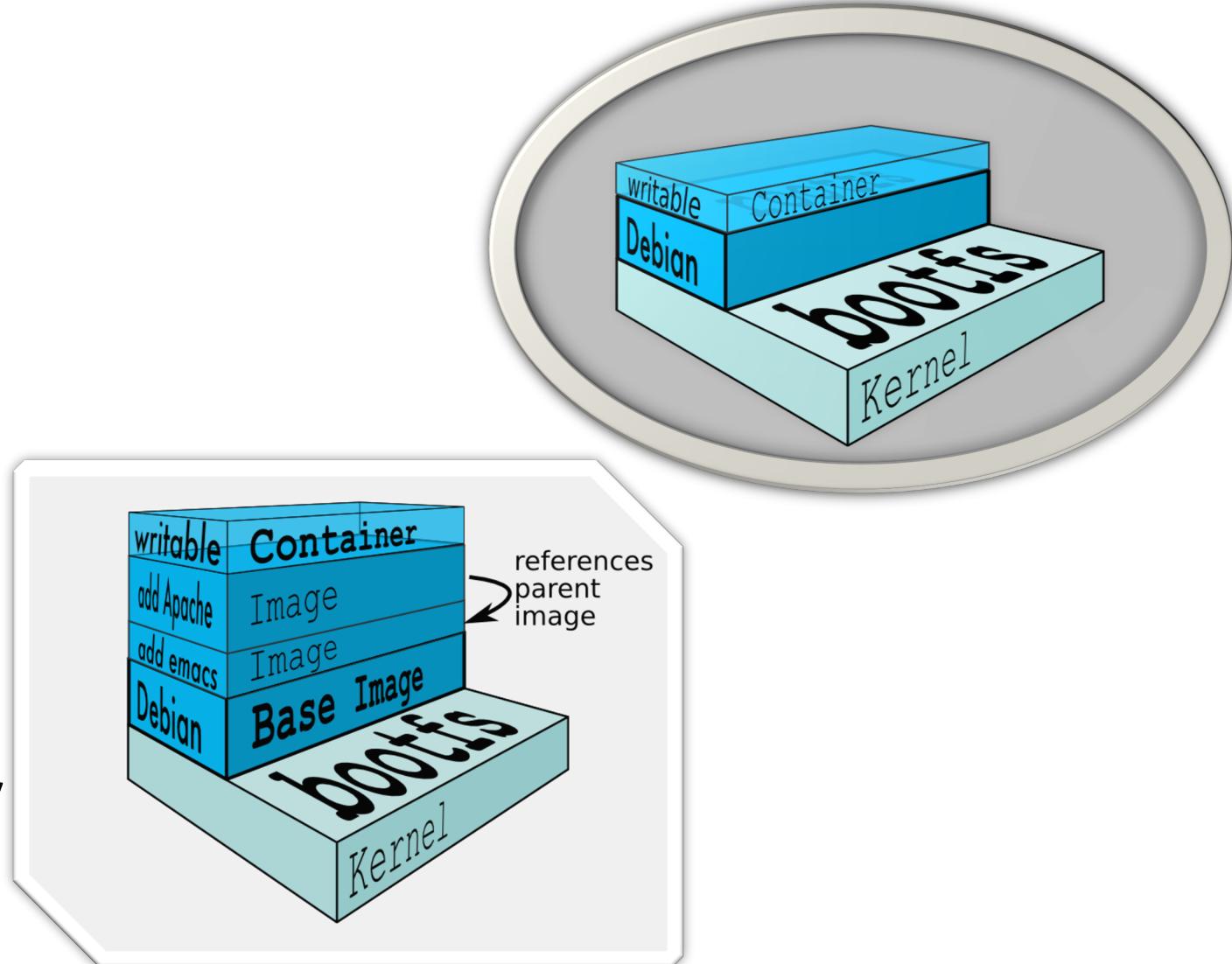
- Docker Engine
  - CLI
  - Docker Daemon
  - Docker Registry
- Docker Hub
  - Cloud service
    - Compartir Aplicaciones
    - Automatitzar workflows
    - Composar apps a partir de components
- Docker images
- Docker containers



# Docker images



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

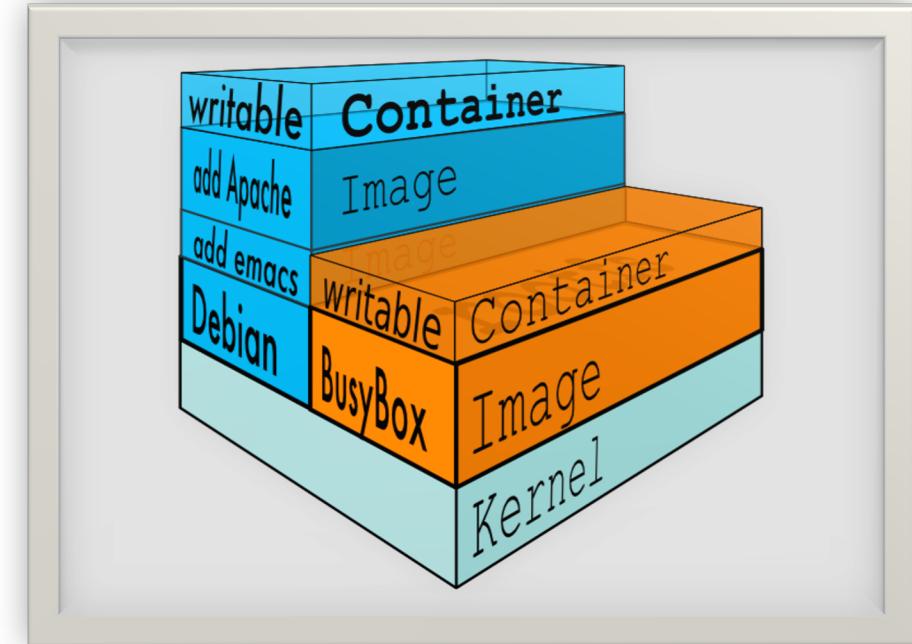


# Docker container



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 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 container 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
```

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

# Dockerfile: RUN

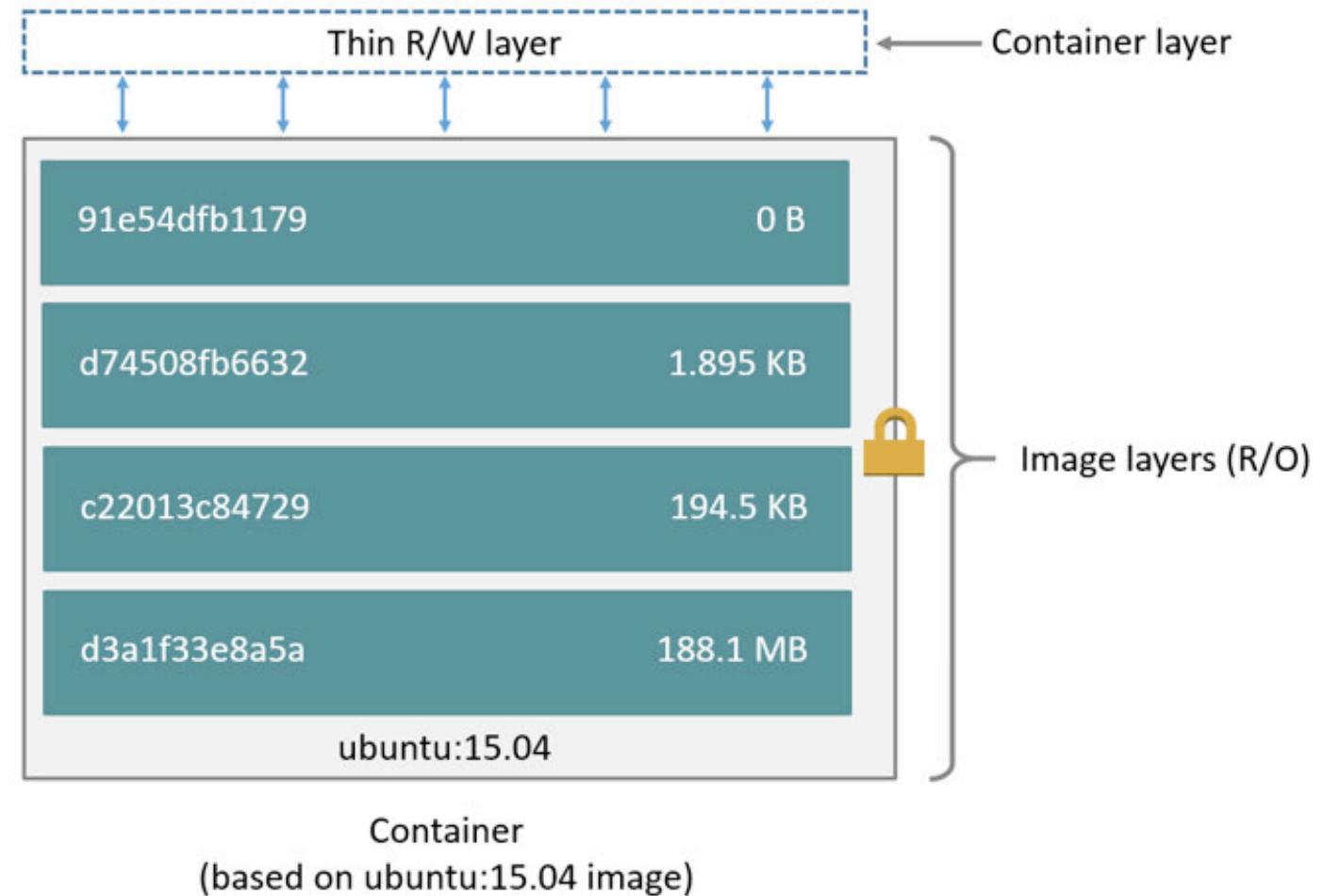


```
FROM ubuntu:18.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

acme/my-base-image:1.0

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

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



# Preguntes típiques

# 5 preguntas típicas 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 preguntas típicas sobre Docker



## 5. Deployments **replace** instead of **update**

***“Immutable infrastructure”***

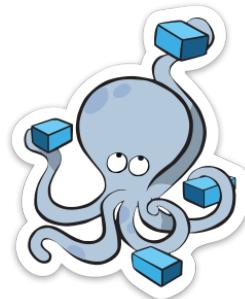
### **Website**

**Update** your app using  
Web Deploy or CI/CD

### **Docker**

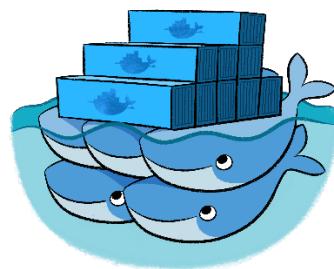
**Replace** running  
containers using CI,  
Don't update the old  
container

# 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 frovlad/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

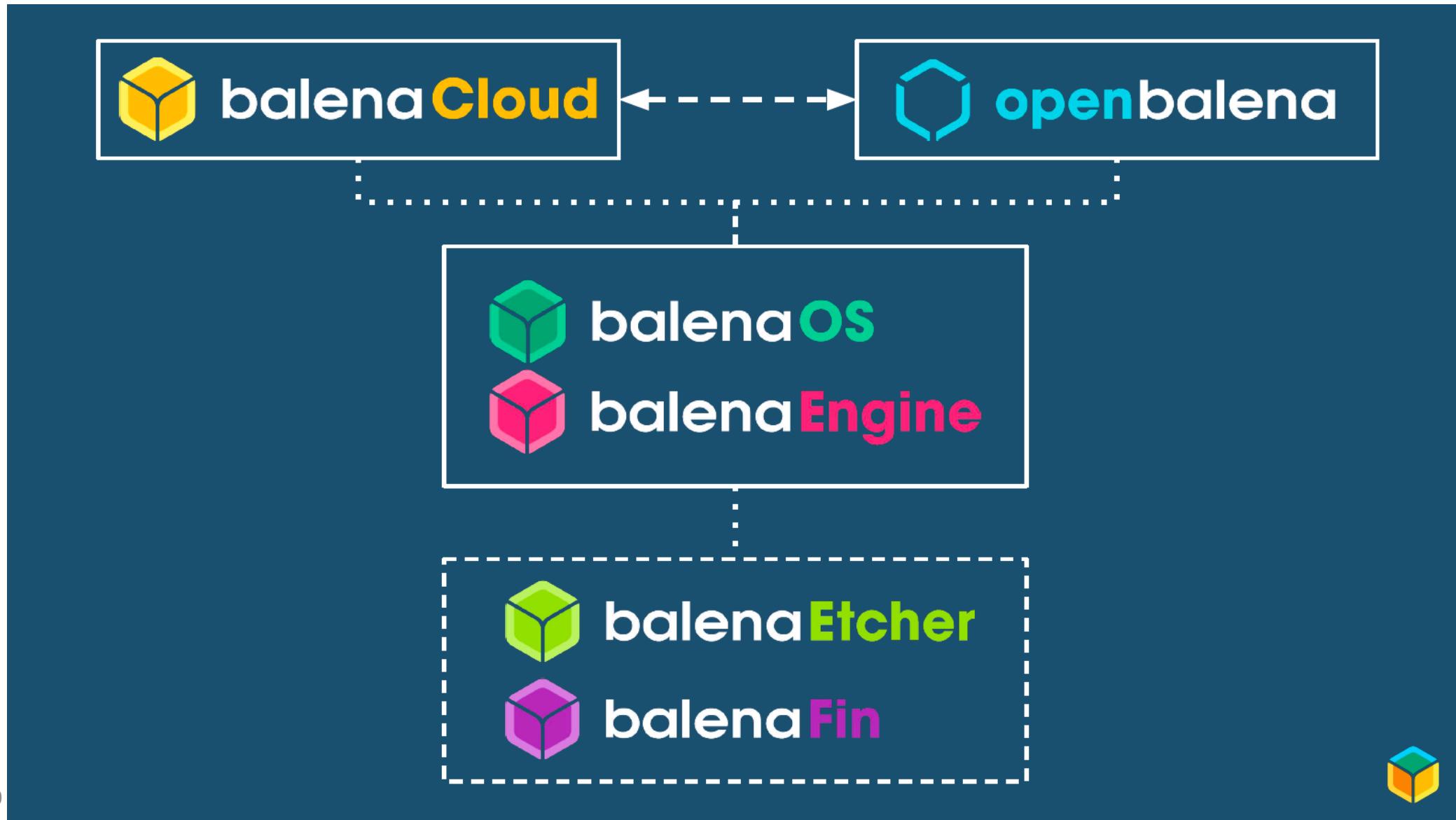
pipelines:
  default:
    - step:
        script:
          - scons
```

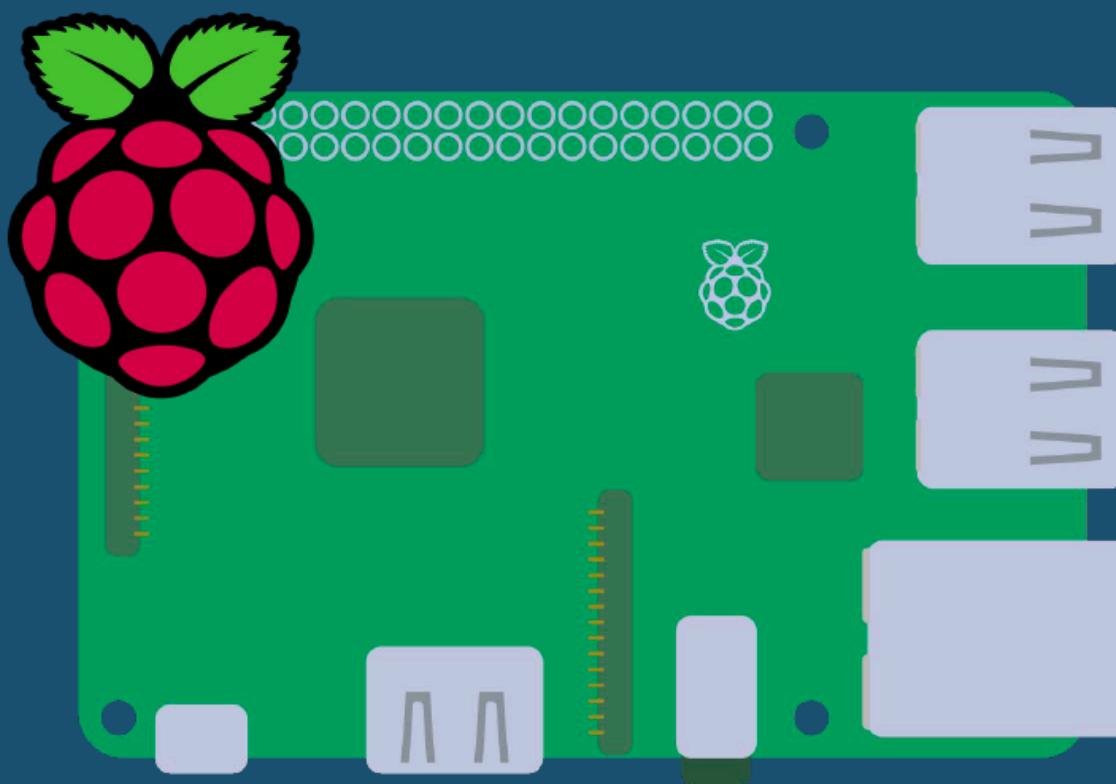
<https://blog.feabhas.com/2017/12/introduction-docker-embedded-developers-part-4-reducing-docker-image-size/>

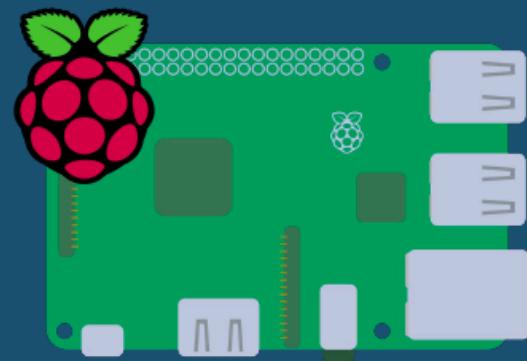
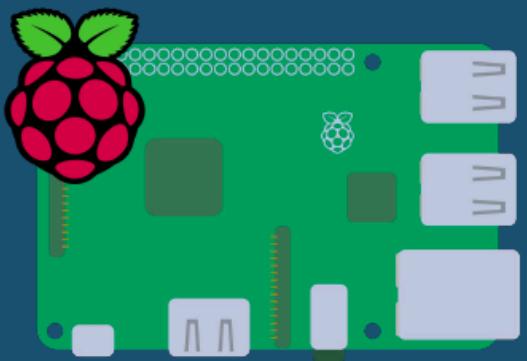
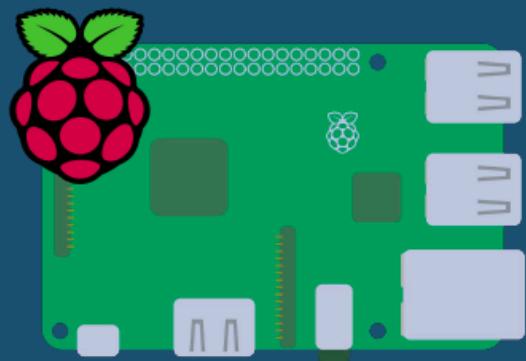
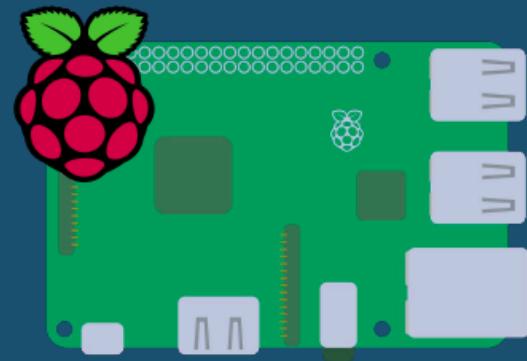
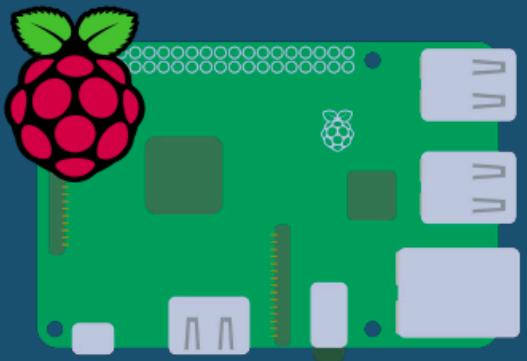
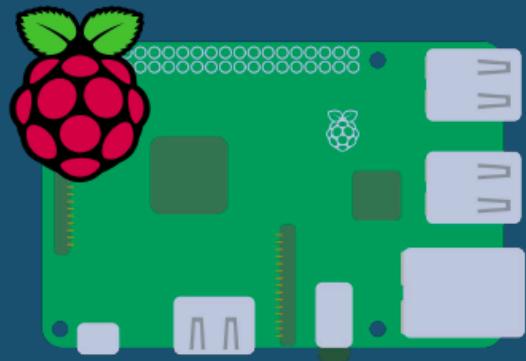
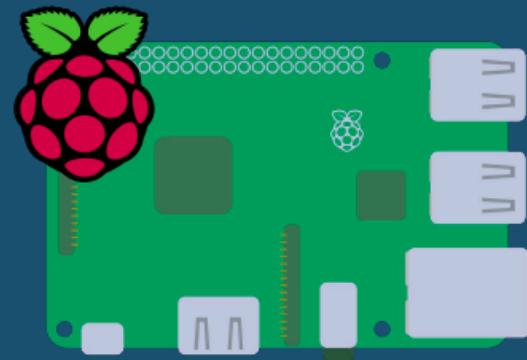
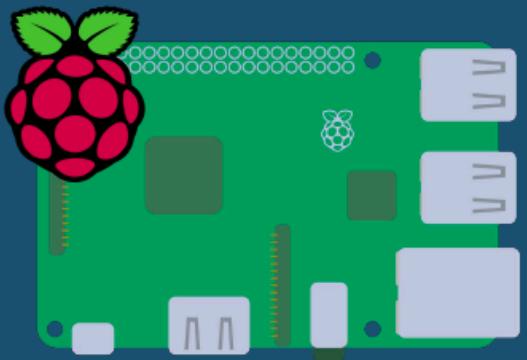
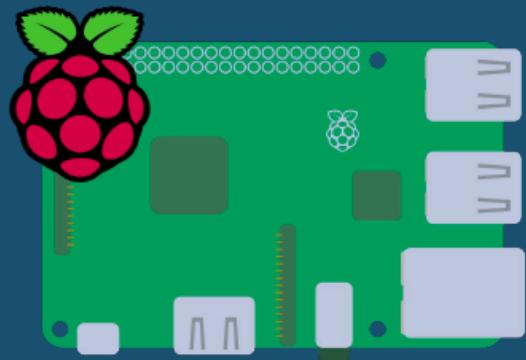
# Run Docker containers on embedded devices

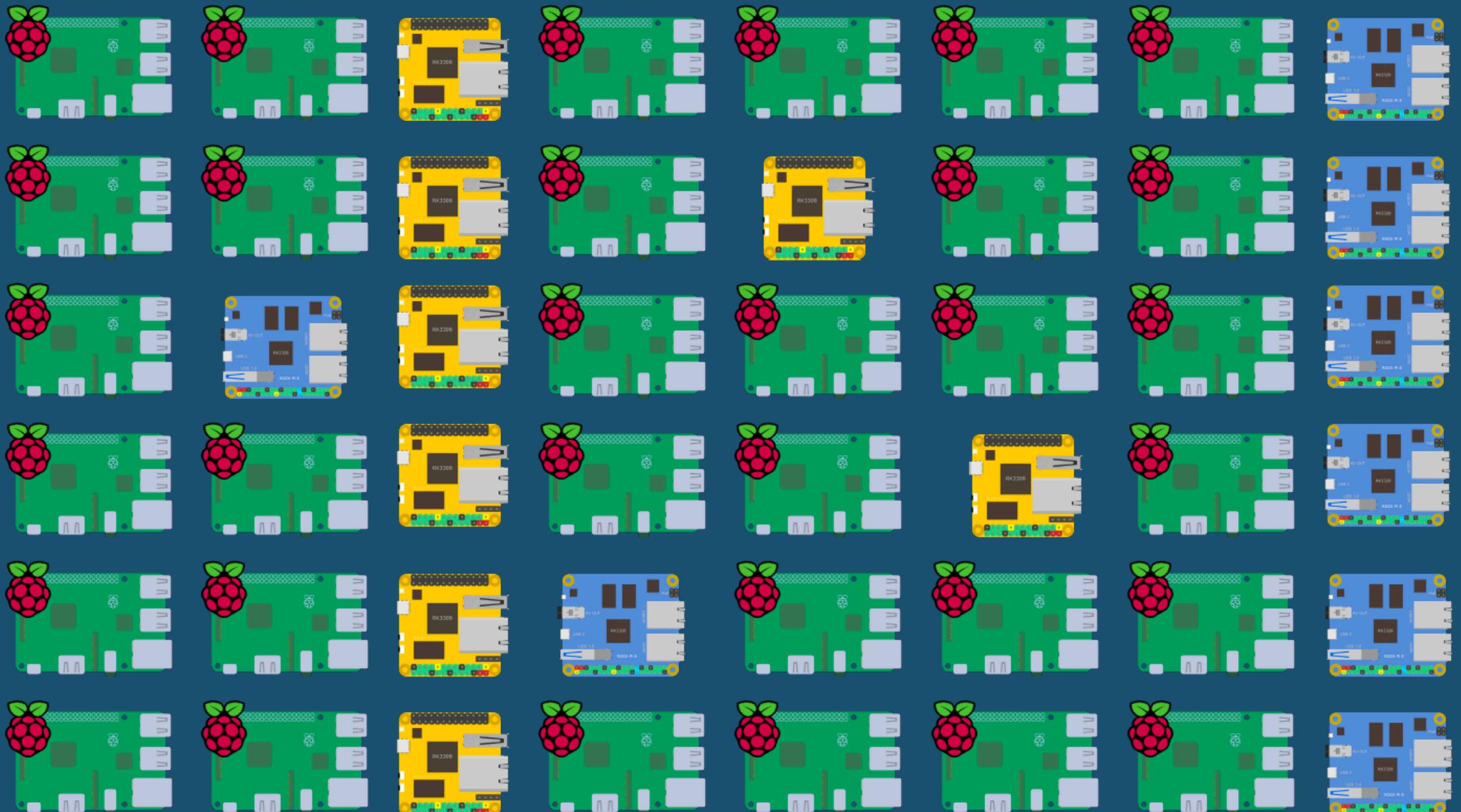
A host OS tailored for containers, designed for reliability,  
proven in production.

[Try balenaOS](#)











[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.



Your first 10 devices are always  
free and full-featured.

[Get started](#)

## Altres

**aws**

Póngase en contacto

Productos Soluciones Precios Documentación Más información Red de socios AWS Marketplace

AWS IoT Información general Servicios de IoT Soluciones de IoT Socios

### AWS IoT

Servicios de IoT para soluciones como industriales y para consumidores

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

**Microsoft Azure**

Contact Sales

Overview Solutions Products Documentation Pricing Training Marketplace Partners Support

### Altres

Simplify IoT development and data transformation with your Azure IoT solution

thethings.io

Connect your hardware and track its everything.

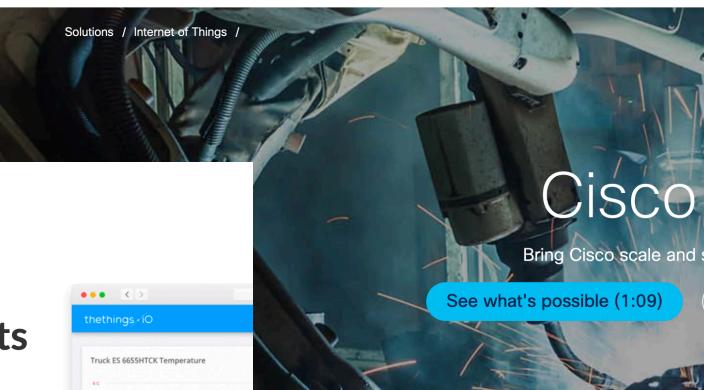
The most simple enterprise IoT platform ready for your company.

**Cisco**

Solutions / Internet of Things /

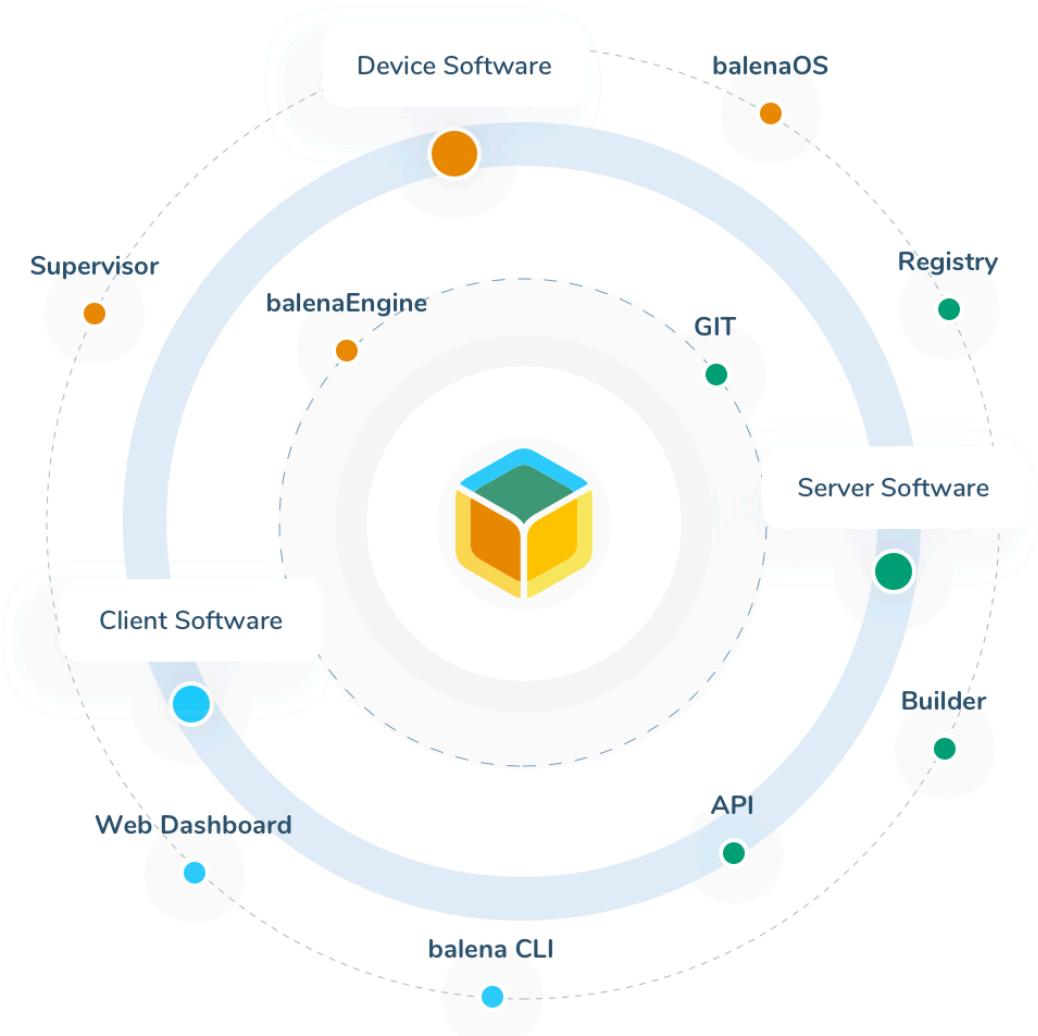
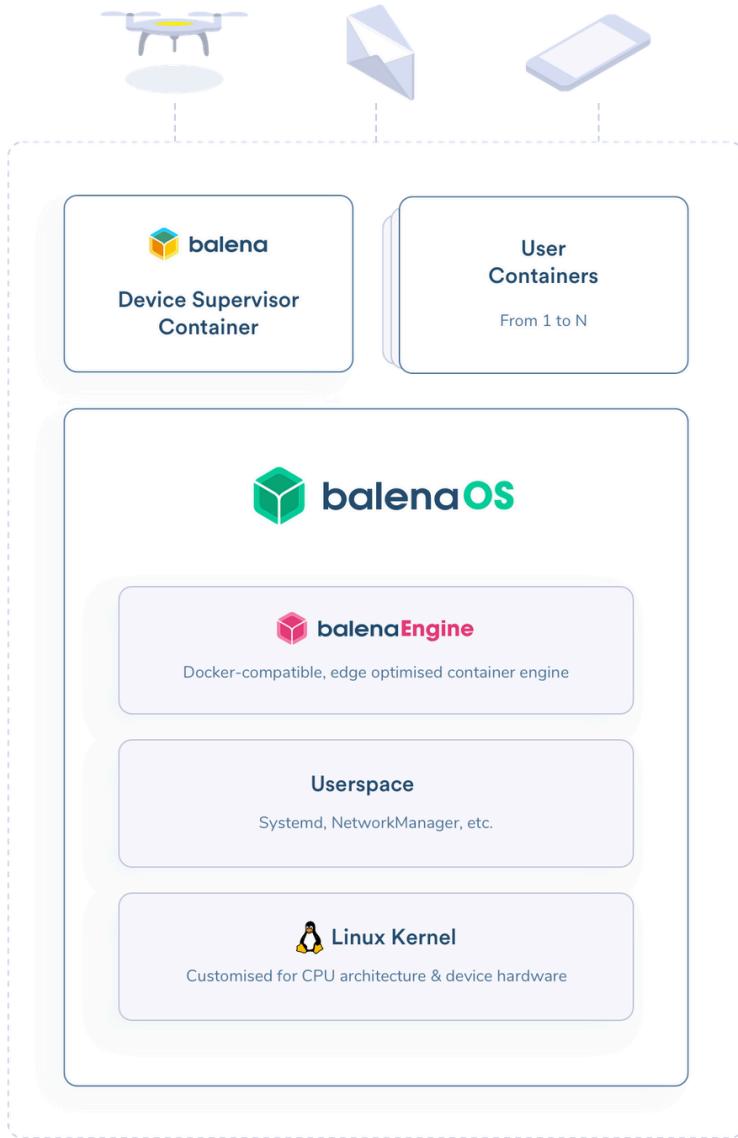
Bring Cisco scale and security to your IoT projects

See what's possible (1:09)

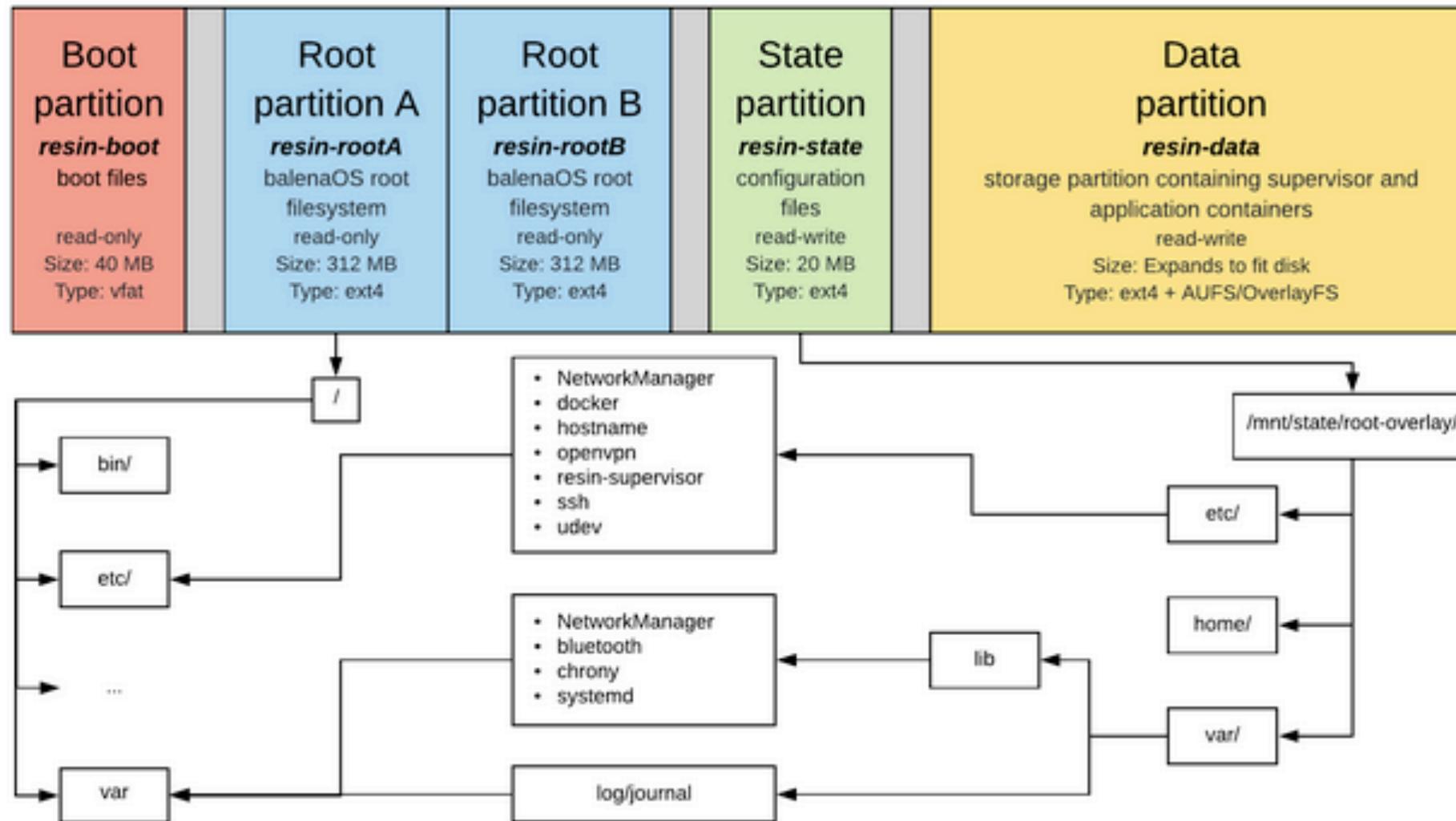


# BalenaOS

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



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



## 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  $\geq$  2.45.0 store a maximum of 32 MB of persistent logs in the data partition of the device.

# BalenaOS

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



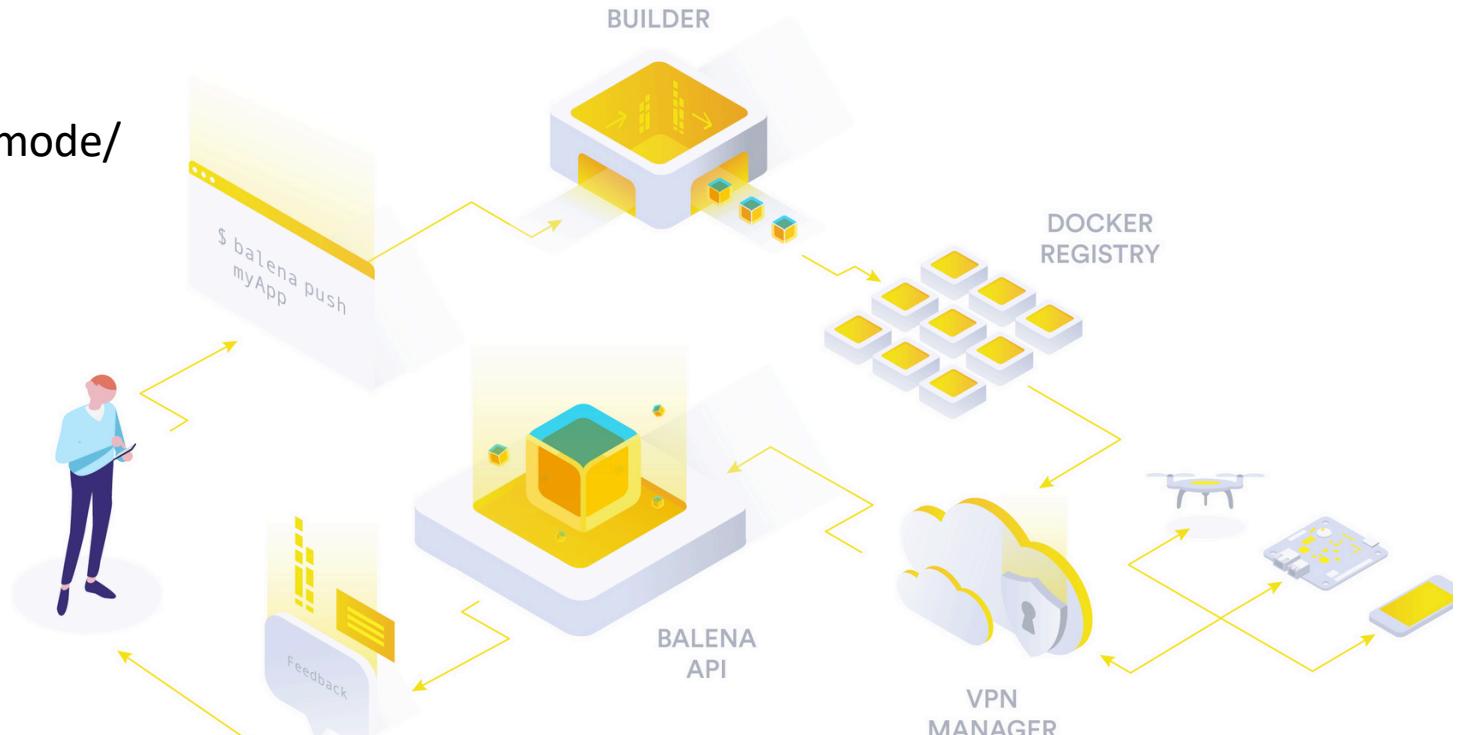
## Development

<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"]
```

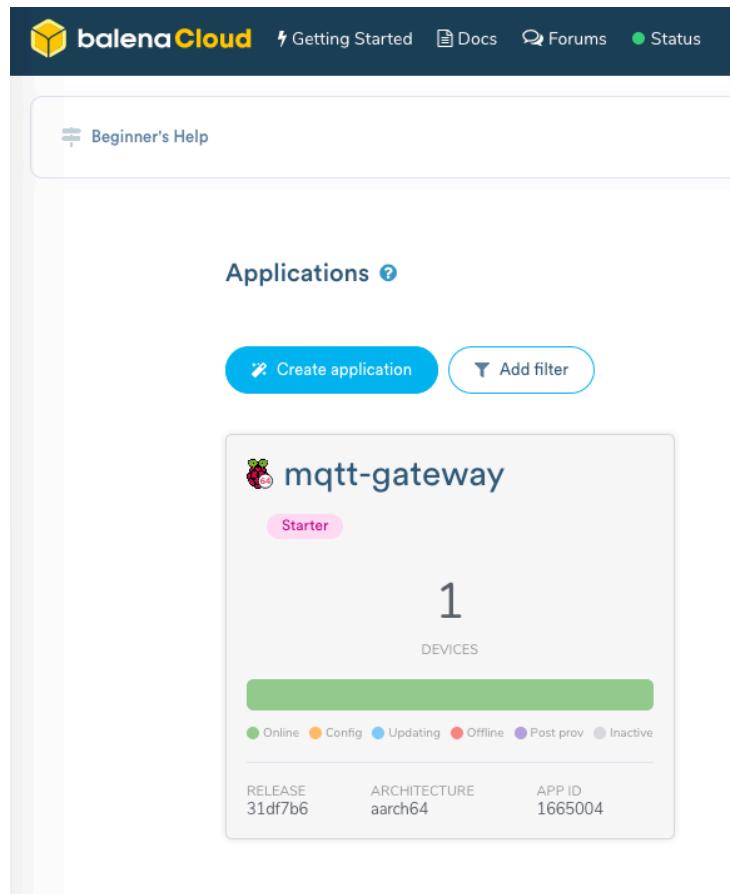


## Deployment

```
$ 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
```

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

## 1) Crear una compta a BalenaCloud



The screenshot shows the balenaCloud interface. At the top, there's a navigation bar with links for 'Getting Started', 'Docs', 'Forums', and 'Status'. Below that is a 'Beginner's Help' section. The main area is titled 'Applications' and shows a single application named 'mqtt-gateway'. This application is labeled as 'Starter' and has 1 device connected. A green progress bar indicates 100% completion. Below the progress bar is a legend for device status: Online (green), Config (orange), Updating (blue), Offline (red), Post prov (purple), and Inactive (grey). At the bottom of the application card, it shows the release '31df7b6', architecture 'arch64', and app ID '1665004'.

## 2) Seguirem les passes...

✓ Create application → ✓ Add device → ✓ Add release → ✓ Add member → What's next?

# Setup BalenaOS :: Crear la Aplicació



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

Create application

Application Name

Default Device Type ?

 Raspberry Pi 3 (using 64bit OS)

Application Type View docs

Starter recommended

[Cancel](#) [Create new application](#)

# Setup BalenaOS :: Afegir dispositius



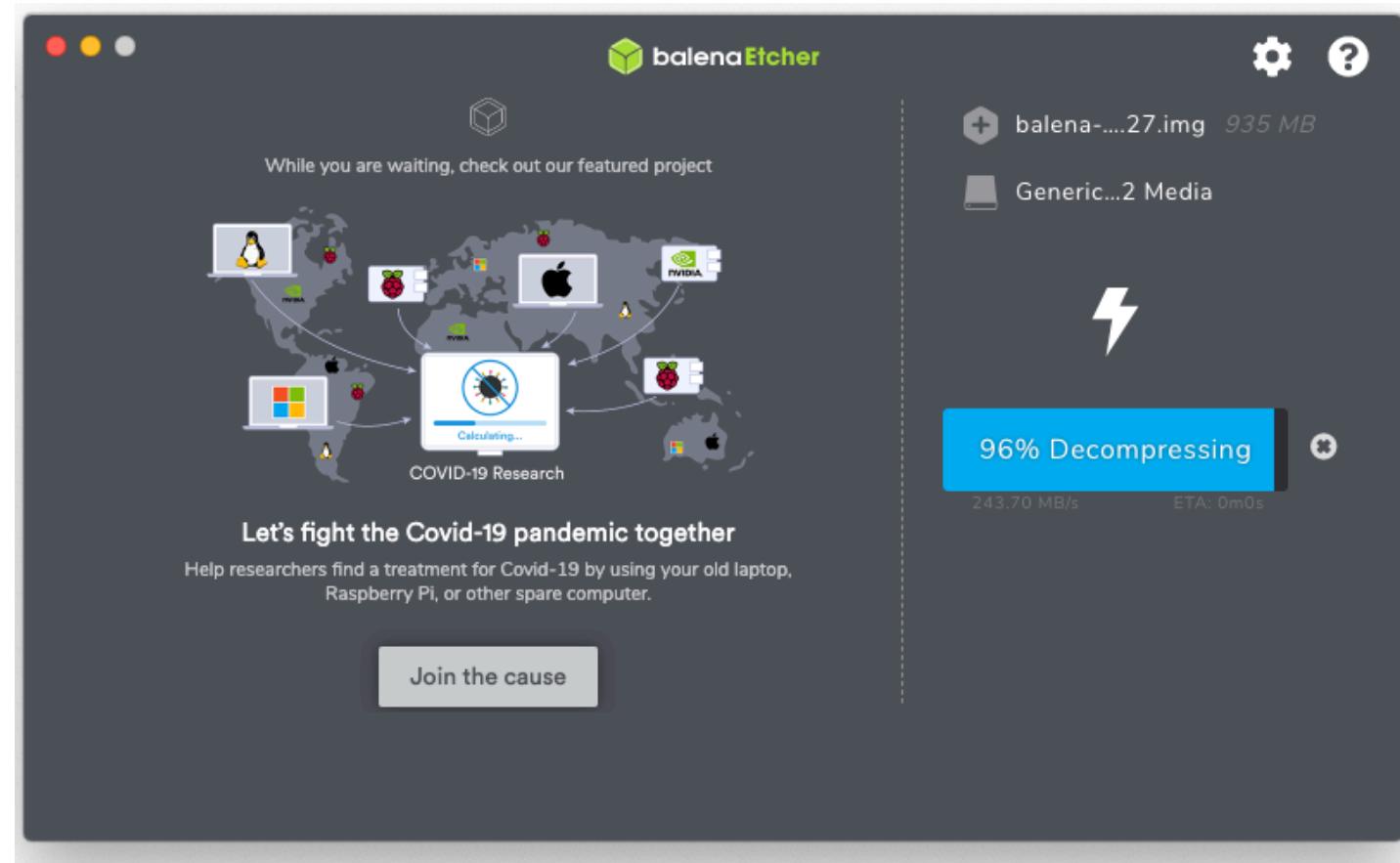
<https://www.balena.io/docs/learn/getting-started/raspberry3/python/>

A screenshot of the balenaCloud web interface. The top navigation bar includes links for 'Getting Started', 'Docs', 'Forums', and 'Status'. On the left, a sidebar menu has 'Devices' selected, indicated by a dark blue background. Other options include 'Fleet Configuration', 'Environment Variables', and 'Service Variables'. The main content area shows the 'mqtt-gateway' application under 'Applications'. A message says 'You don't have any devices yet. How about adding one?' with a dashed arrow pointing to a blue button labeled '+ Add device'. There are also 'Add filter' and 'Search entries...' buttons.

# 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/raspberry3/python/>

The screenshot shows the balenaCloud web interface. On the left, a sidebar menu includes: Home, Devices (selected), Fleet Configuration, Environment Variables, Service Variables, and Releases (selected). The main content area shows a list of devices under the application "mqtt-gateway". One device, "patient-field", is listed with status "Online", name "patient-field", last seen "Online (for a few seconds)", UUID "a05984e", and OS Version "Unknown". A modal window is open over the list, titled "Add release", with a message: "You don't have any releases yet. How about adding one?". At the top of the page, there are navigation links: Getting Started, Docs, Forums, and Status. A progress bar at the top right indicates a workflow: Create application → Add device → Add release.

# 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 ./

Applications > 🍄 mqtt-gateway > patient-field

## Device Service Variables ?

+ Add variable

| Service | Name   |
|---------|--------|
| gateway | HOSTDB |
| gateway | PORTDB |

25.5.2020



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



balenaCloud Getting Started Docs Forums Status David Raba DR ▾

Back Beginner's Help Create application Add device Add release Add member What's next?

Summary Applications > mqtt-gateway > patient-field

DEVICE

**patient-field**

STATUS: Online (a05984e)

UUID: a05984e

TYPE: Raspberry Pi 3 (using 64bit OS)

LAST ONLINE: Online (for 36 minutes)

HOST OS VERSION: balenaOS 2.47.0+rev1 (production)

SUPERVISOR VERSION: 10.6.27

CURRENT RELEASE: 425ac83

TARGET RELEASE: 425ac83

IP ADDRESS: 192.168.0.29

PUBLIC DEVICE URL: [View](#)

TAGS (0): No tags configured yet

NOTES: Add device notes...

SERVICES

| Service | Status  | Release |
|---------|---------|---------|
| gateway | Running | 425ac83 |
| mqtt    | Running | 425ac83 |

Logs

UTC Timestamps

Search entries... Views

Add filter

|                                   |   |
|-----------------------------------|---|
| 19.05.20 14:18:20 (+0200) gateway | Saved in Mongo document ID Sec3ce8c415a98576b197427 |
| 19.05.20 14:18:36 (+0200) gateway | Rx MQTT   |
| 19.05.20 14:18:36 (+0200) gateway | Saving  |
| 19.05.20 14:18:36 (+0200) gateway | Storing   |
| 19.05.20 14:18:36 (+0200) gateway | Saved in Mongo document ID Sec3ce9c415a98576b197428 |
| 19.05.20 14:20:53 (+0200) gateway | Rx MQTT   |
| 19.05.20 14:20:53 (+0200) gateway | Saving  |
| 19.05.20 14:20:53 (+0200) gateway | Storing   |
| 19.05.20 14:20:53 (+0200) gateway | Saved in Mongo document ID Sec3cf25415a98576b197429 |

Terminal

| Host OS   | + |
|---|---|
| 4650 root 0 IW< [kworker/1:0H]  |   |
| 4694 root 0 IW [kworker/2:0]  |   |
| 4712 root 0 IW [kworker/3:0-eve]  |   |
| 4746 root 6140 R sshd: root@pts/1   |   |
| 4754 root 3600 S /bin/bash -l   |   |
| 4763 root 3428 R ps   |   |
| root@a05984e:~# ps   grep mqtt  |   |
| 4766 root 3256 S grep mqtt  |   |
| root@a05984e:~# ps   grep 1883  |   |
| 4054 root 695m S /usr/bin/balena-engine-proxy -proto tcp -host-ip 0.0.0.0 -host-port 1883 |   |
| 4768 root 3256 S grep 1883  |   |
| root@a05984e:~#   |   |

# Setup BalenaOS :: Simularem un missatge MQTT



balenaCloud

Getting Started Docs Forums Status

Back Beginner's Help Create application

Summary

Applications > mqtt-gateway > patient-field

DEVICE

**patient-field**

STATUS: Online (UUID: a05984e)

TYPE: Raspberry Pi 3 (using 64bit)

LAST ONLINE: Online (for 36 minutes)

HOST OS VERSION: balenaOS 2.47.0+rev1 (production)

SUPERVISOR VERSION: 10.6.27

CURRENT RELEASE: 425ac83

TARGET RELEASE: 425ac83

IP ADDRESS: 192.168.0.29

PUBLIC DEVICE URL: [View](#)

TAGS (0): No tags configured yet

NOTES: Add device notes...

SERVICES

| Service | Status  | Release |
|---------|---------|---------|
| gateway | Running | 425ac83 |
| mqtt    | Running | 425ac83 |

<http://mqtt-explorer.com/>

MQTT Explorer

192.168.0.29

\$SYS (46 topics, 653 messages)

sensors = {'humidity':40.3}

actuators = {'light':1}

Value

Comparing with selected message: + 1 line, - 1 line

QoS: 0  
19/05/2020 14:18:36

- {'humidity':34.3}  
+ {'humidity':40.3}

History

19/05/2020 14:18:36  
{'humidity':40.3}

19/05/2020 14:18:20 (-15.87 seconds)  
{'humidity':34.3}

Publish

Topic: actuators

raw xml json

{'light':1}

PUBLISH

# Setup BalenaOS :: Revisem la BBDD



Robo 3T - 1.3

Welcome db.getCollection('sensors').find()

Master IoT edu.insylo.io:47017 MasterIoT

db.getCollection('sensors').find({})

sensors 0.03 sec.

| Key                                   | Value                                | Type     |
|---------------------------------------|--------------------------------------|----------|
| 1 ObjectId("5ec3ce8c415a98576b19...") | { 6 fields }                         | Object   |
| _id                                   | ObjectId("5ec3ce8c415a98576b197427") | ObjectId |
| topic                                 | sensors                              | String   |
| payload                               | {'humidity':34.3}                    | String   |
| qos                                   | 0                                    | Int32    |
| timestamp                             | 1589890700                           | Int32    |
| datetime                              | 19/05/2020 12:18:20                  | String   |
| 2 ObjectId("5ec3ce9c415a98576b19...") | { 6 fields }                         | Object   |
| _id                                   | ObjectId("5ec3ce9c415a98576b197428") | ObjectId |
| topic                                 | sensors                              | String   |
| payload                               | {'humidity':40.3}                    | String   |
| qos                                   | 0                                    | Int32    |
| timestamp                             | 1589890716                           | Int32    |
| datetime                              | 19/05/2020 12:18:36                  | String   |
| 3 ObjectId("5ec3cf25415a98576b19...") | { 6 fields }                         | Object   |
| _id                                   | ObjectId("5ec3cf25415a98576b197429") | ObjectId |
| topic                                 | actuators                            | String   |
| payload                               | {'light':1}                          | String   |
| qos                                   | 0                                    | Int32    |
| timestamp                             | 1589890853                           | Int32    |
| datetime                              | 19/05/2020 12:20:53                  | String   |

Logs

<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/>

# Referencies – BalenaOS / Docker



- How to burn image with python based environment
  - <https://www.balena.io/docs/learn/getting-started/raspberry3/python/>
- Hello world Python service
  - <https://github.com/balena-io-examples/balena-python-hello-world>
- Integració amb plataforma de injecció 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/>