

DEPLOYMENT MÚLTIPLES DISPOSITIUS AMB RESIN.IO I DOCKER

O. Vazquez

19 juny 2019

Índex

1. Objectiu.....	3
2. Solució adoptada.....	3
3. Hardware Utilitzat.....	4
4. Software Utilitzat.....	5
5. Desenvolupament del projecte.....	10
A. Configurar Balena.io el SO a la Raspberry.....	10
B. Inicialització a Docker de Git vinculat a BalenaOS del nostre device.....	13
C. Com enviar un codi Python per que s'executi directament a la Raspberry només iniciar el contenidor.....	15
D. Afegir nous dispositius a la nostra aplicació.....	17
6. Bibliografia utilitzada.....	18

1. Objectiu

L'objectiu del projecte consisteix en posar en pràctica una solució de desplegament de configuracions i aplicacions de forma simultània a múltiples dispositius IOT. En aquest cas Raspberries.

Aquesta solució ens permetrà simplificar el desplegament de software, ja sigui en la configuració inicial o posteriors actualitzacions de forma simultània a diferents dispositius independentment de la seva ubicació.

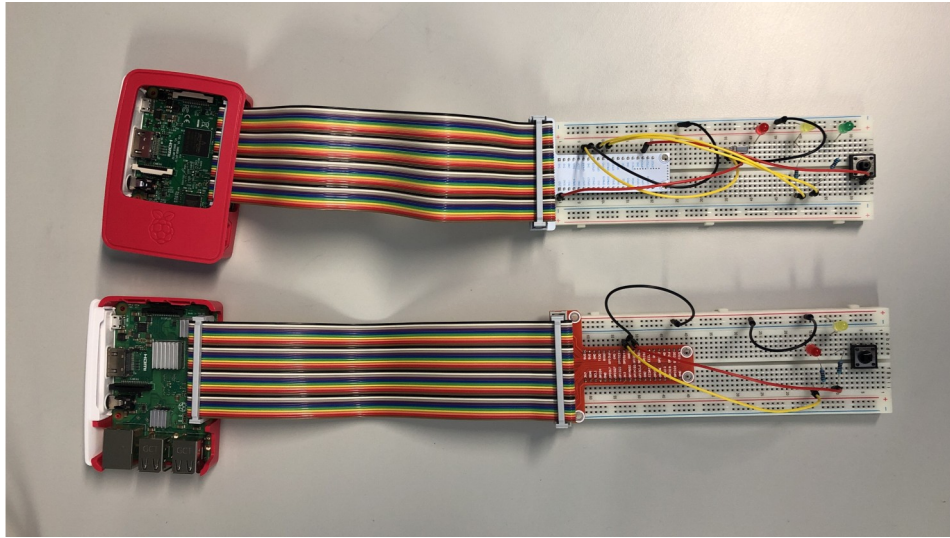
Simplifica també una futura ampliació de la infraestructura IOT en el cas que s'afegeixin nous equips IOTs a la xarxa inicial.

2. Solució adoptada

La solució escollida es basa en la plataforma Balena.io (RESIN.io anteriorment) per l'activació i gestió del serveis i equips que volem desplegar i la plataforma Docker per la construcció de contenidors que seràn els que portaran la configuració i software que volem executar a la xarxa d'equipament IOT desplegada.

3. Hardware Utilitzat

- 2 Raspberryies 3B+
- 2 Targetes SDCARD i adaptador a microSDCARD
- 2 BreadBoard OSOYOO
- 2 GPIO Extension Board
- 4 LEDs
- 2 cables Ribbon 40 pines
- 6 cables jumpers per protoboard (mascle-mascle)
- 4 resistències 200K



4. Software Utilizat

1. Instal.lar Balena Etcher

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

2. Crear un compte a Balena.io

<https://balena.io/>

3. Cremar Balena OS a la nostra Raspberry després de generar una imatge.

3. Instal.lar Docker ToolBox (+ Docker Quickstart Terminal)

<https://software.intel.com/en-us/intel-system-studio-docker-install-windows-install-docker-toolbox>

Un cop instal·lat caldrà inicialitzar el Docker Quickstart Terminal:

[illegible]

Provem que la instal·lació i inicialització de Docker funciona correctament. Per això executem l'ordre: `docker run hello-world`:

```
MINGW64: c:/Program Files/Docker Toolbox
Start interactive shell
Alumne@DESKTOP-LEG3D3B MINGW64 /c/Program Files/Docker Toolbox
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:41a65640635299bab090f783209c1e3a3f11934cf7756b09cb2f1e02147c6ed8
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

A continuació, farem una introducció a la sintaxi bàsica a utilitzar a Docker:

FROM

La instrucció FROM inicialitza un nou build i configura la imatge base per les posteriors instruccions. Tot Dockerfile ha de començar sempre amb aquesta instrucció. Sempre que sigui possible utilitzar imatges oficials i actuals. La sintaxi seria una de les següents opcions:

```
FROM <image> [AS <name>]
FROM <image>[:<tag>] [AS <name>]
FROM <image>[@<digest>] [AS <name>]
```

Opcionalment se li pot assignar un nom a la construcció de la nostra imatge simplement afegint AS a la instrucció. El nom es podrà fer servir en posteriors instruccions FROM i COPY per referir-se a la imatge construïda.

S'hauria d'afegir `--from=<name|index>` a la instrucció.

Els valors tag o digest són opcionals. Si els omitim el constructor (builder) assumeix el darrer tag per defecte. Si el constructor no troba el valor tag, retorna un error.

RUN

Combinar sempre RUN `apt-get` amb `apt-get install` a la mateixa instrucció RUN. Per exemple:

```
RUN apt-get update && apt-get install -y \  
package-bar \  
package-baz \  
package-foo
```

Utilitzar `apt-get update` en una instància RUN independent origina problemes en les següents instruccions `apt-get install`. Docker veu que les instruccions inicials i posteriors són idèntiques i reutilitza la caché utilitzada anteriorment. Com a conseqüència, el `apt-get update` no s'executa perquè BUILD utilitza la versió caché, i com que llavors `apt-get update` no està activa, es poden obtenir versions desfasades dels paquets que volem instal·lar.

Utilitzar `RUN apt-get update && apt-get install -y` assegura que el fitxer Dockerfile instal·la la darrera versió dels paquets sense necessitat d'una intervenció manual posterior. Aquesta tècnica es coneix com “cache busting”.

A continuació un exemple d'una bona estructura de la instrucció RUN:

```
RUN apt-get update && apt-get install -y \  
aufs-tools \  
automake \  
build-essential \  
curl \  
dpkg-sig \  
libcap-dev \  
libsqlite3-dev \  
mercurial \  
reprepro \  
ruby1.9.1 \  
ruby1.9.1-dev \  
s3cmd=1.1.* \  
&& rm -rf /var/lib/apt/lists/*
```

CMD

La instrucció CMD hauria de ser utilitzada per executar el software contingut a la imatge, acompanyada dels arguments necessaris. CMD s'hauria de fer servir en alguna

de les dues següents formes:

`CMD ["executable", "param1", "param2"...]`. Aquesta forma de la instrucció és la recomenada per qualsevol servei basat en imatge. Ex: `CMD["Python3", "aa.py"]`

`CMD ["executable1", "executable 2", ...]`. Aquesta forma retornaria una shell interactiva llesta per utilitzar. Ex: `CMD["python3"]`

ENV

Per fer que el nou software sigui més còmode i fàcil d'utilitzar, podem utilitzar ENV per actualitzar les variables d'entorn PATH a les instal·lacions del contenidor. Per exemple:

`ENV PATH /usr/local/nginx/bin:$PATH` assegura que `CMD ["nginx"]` funcioni directament.

La instrucció ENV també és útil per proveir de les variables d'entorn requerides específicament pels serveis que volem posar en un contenidor, per exemple PGDATA Postgres.

Finalment, ENV es pot utilitzar per fixar els números de les versions que utilitzem de forma que sigui més senzill de mantenir. Per exemple:

```
ENV PG_MAJOR 9.3
ENV PG_VERSION 9.3.4
RUN curl -SL http://example.com/postgres-$PG_VERSION.tar.xz | tar -xJC
  /usr/src/postgress && ...
ENV PATH /usr/local/postgres-$PG_MAJOR/bin:$PATH
```

De forma similar a utilitzar variables constants en un programa enlloc de fer servir valors directament, aquesta funcionalitat permet canviar amb una simple instrucció ENV la versió del software del nostre contenidor.

Cada línia ENV crea una nova capa intermitja, igual que les comandes RUN. Això significa que encara que desconfigures les variables d'entorn en una capa posterior, persistiràn a la seva capa i el seu valor es podrà modificar.

WORKDIR

Per simplicitat i fiabilitat, hauríem d'utilitzar sempre rutes absolutes pel WORKDIR. A

més, hauríem d'utilitzar WORKDIR en lloc d'instruccions com:

`RUN cd ... && do-something`, les quals són complicades de llegir, analitzar i mantenir.

ADD i COPY

Encara que ADD i COPY tenen un funcionament similar, normalment COPY és el més utilitzat. Això és per que és més transparent que ADD.

COPY únicament suporta el copiat bàsic d'arxius locals al contenidor, mentre que ADD té més funcionalitats (com extracció local de tar i suport URL remot). Per aquest motiu, el millor ús d'Add es per l'autoextracció de fitxers tar a la imatge. Ex: `ADD rootfs.tar.xz /`.

Si tenim múltiples passes al nostre Dockerfile que utilitzen diferents arxius segons el seu context, farem COPY individual d'ells enlloc de fer un COPY de tots ells alhora.

Per exemple:

```
COPY requirements.txt /tmp/
RUN pip install --requirement /tmp/requirements.txt
COPY . /tmp/
```

Resulta en menys invalidacions de cache per la línia RUN que si fem un `COPY . /tmp/` abans.

Com que la mida de la imatge importa, utilitzar ADD per extraure paquets des de URLs remotes no seria aconsellable, s'hauria d'utilitzar millor curl o wget. D'aquesta forma es poden eliminar els fitxers que no es necessiten després que s'han estret i d'aquesta forma no cal afegir més capes a la imatge. Per exemple, hauríem d'evitar fer coses com la següent:

```
ADD http://example.com/big.tar.xz /usr/src/things/
RUN tar -xJf /usr/src/things/big.tar.xz -C /usr/src/things
RUN make -C /usr/src/things all
```

I en el seu lloc, fer alguna cosa així:

```
RUN mkdir -p /usr/src/things \
    && curl -SL http://example.com/big.tar.xz \
    | tar -xJC /usr/src/things \
    && make -C /usr/src/things all
```

Per altres coses (arxius, directoris) que no requereixen ADD tar amb funcionalitat autoextracció, s'hauria de fer servir sempre COPY.

5. Desenvolupament del projecte

A. Configurar Balena.io el SO a la Raspberry

1. Generar parella de claus pública i privada a Docker QuickStart Terminal.

```
ssh-keygen
```

si no es modifica el nom dels documents, es generaran dos arxius:

```
id_rsa
```

```
id_rsa.pub
```

2. Anar a la ruta on s'han guardat les claus.

```
/c/Users/Alumne/.ssh/
```

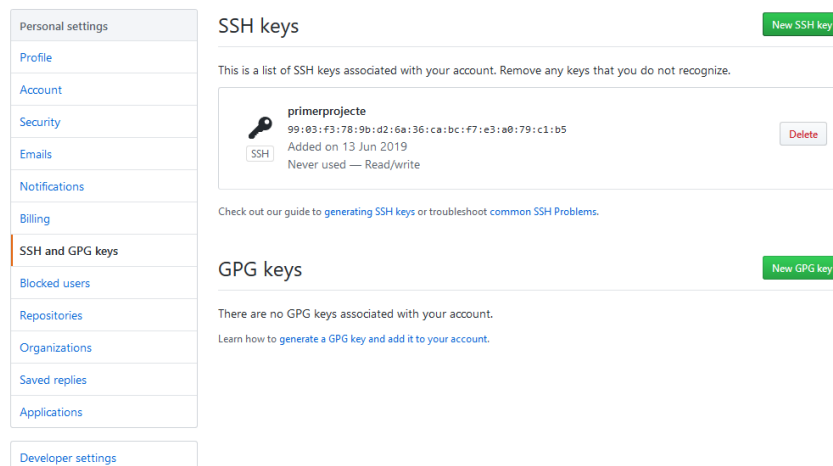
3. Copiar la clau pública fent

```
cat id_rsa.pub
```

i copiar-la en un fitxer text.

Aprofitarem per copiar-la al nostre compte de GitHub:

La copiarem com a nova clau a l'apartat SSH and GPG Keys de la configuració del nostre compte:



The screenshot shows the GitHub 'SSH and GPG keys' settings page. On the left is a sidebar with navigation links: Personal settings, Profile, Account, Security, Emails, Notifications, Billing, SSH and GPG keys (highlighted), Blocked users, Repositories, Organizations, Saved replies, Applications, and Developer settings. The main content area is titled 'SSH keys' and includes a 'New SSH key' button. It contains a list of SSH keys with one key named 'primerprojecte' (added on 13 Jun 2019, never used, read/write permissions) and a 'Delete' button. Below the list is a link to a guide on generating SSH keys. The 'GPG keys' section below it shows 'There are no GPG keys associated with your account.' and a 'New GPG key' button, along with a link to learn how to generate a GPG key.

4. Crear un compte a RESIN.IO

<https://dashboard.balena-cloud.com/signup>

5. Modificar a les preferències del compte les claus SSH:

Entrar la clau manualment, pegant la clau pública que hem recollit a l'apartat 3.

Add a new key ?

Title

Add a title for your key

Key

Enter your key

Cancel Add SSH key

Account details SSH keys Access tokens Profile

IMPORT FROM GITHUB ENTER SSH KEY MANUALLY

<input type="checkbox"/>	Name	Fingerprint
<input type="checkbox"/>	primerprojecte	99:03:f3:78:9b:d2:6a:36:ca:bc:f7:e3:a0:79:c1:b5

Un cop fet, apareixerà la clau llistada amb el nom que li hem donat:

<input type="checkbox"/>	Name	Fingerprint	Date Added	Key
<input type="checkbox"/>	primerprojecte	99:03:f3:78:9b:d2:6a:36:ca:bc:f7:e3:a0:79:c1:b5	Jun 13th 2019, 12:58 pm	ssh-rsa AAAAB3NzaC1yc2EAAAADAQ...

6. Crear una aplicació nova:

Applications ?

Create application Add filter

Omplim dades: Nom de l'aplicació, Tipus d'equip (Raspberry Pi 3) i tipus d'aplicació (Starter). Creem l'aplicació:

Create application

Application Name

The name should be at least 4 characters long. It can only contain letters and numbers.

Default Device Type ?

Raspberry Pi 3

Application Type View docs


Starter recommended

Cancel CREATE NEW APPLICATION

L'aplicació ja apareix llistada al resum:

Applications ?

Create application Add filter

**Balena1**
Starter


1
DEVICES

Online Config Updating Offline Post prov Inactive

RELEASE	ARCHITECTURE	APP ID
No commit	armv7hf	1476866

7. Afegir un nou dispositiu a l'aplicació:


Cliquem a l'aplicació i li donem al botó d'afegir device:

Applications > **Balena1**
Starter

Add filter

+ Add device

Seleccionem que volem connectar WIFI + ETHERNET i indiquem el nom i password de la xarxa WIFI a la que es connectarà la raspberry.

 Add a new device

SELECT DEVICE TYPE

Raspberry Pi 3

SELECT BALENAOS VERSION

v2.36.0+rev2 (recommended) ☐ Show outdated versions

SELECT EDITION

Development ☒ Production Production images are ready for production deployments, but don't offer easy access for local development.

NETWORK CONNECTION

Ethernet Only ☒ Wifi + Ethernet

WIFI SSID

Cisco032Z

WIFI PASSPHRASE

ADVANCED

The Raspberry Pi 3 is not capable of connecting to 5GHz Wifi networks unless you use an external Wifi adapter that supports it. The Raspberry Pi 3 B+ is capable of connecting to both 5GHz and 2.4GHz networks.

Download balenaOS (~172 MB)

Instructions

- Use the form on the left to configure and download balenaOS for your new device.
- Write the OS file you downloaded to your SD card. We recommend using [Etcher](#).
- Insert the freshly burnt SD card into the Raspberry Pi 3.
- Connect your Raspberry Pi 3 to the internet, then power it up.
- Your device should appear in your application dashboard within a few minutes. Have fun!

For more details please refer to our [getting started guide](#).

Cliquem a descarregar BalenaOS. Amb això el que estarem fent es descarregar la imatge que farem servir per cremar a la Raspberry.

8. Cremem la imatge que ens hem descarregat a la targeta SDCARD que instal.larem a la Raspberry. Farem servir Balena_Etcher.

9. Configurar paràmetres de la connexió WIFI a la imatge que ens hem baixat. Abans de posar la SDCARD a la Raspberry, modifiquem a la carpeta **system-connections**, editar el fitxer **resin-wifi-01**. A l'apartat [ipv4], indicar ip fixa, màscara, gateway, dns i mètode de connexió manual:

address1=10.199.160.10/24,10.199.160.254

dns=8.8.8.8,4.4.4.4;

dns-search=

method=manual

10. Posem la SDCARD a la Raspberry i la connectem.

Veurem que el device apareix a la pàgina de RESIN.IO, llistada com a nou device un cop ha agafat IP i s'ha connectat a internet.

The screenshot shows the Balena.io interface. At the top, there's a header with 'Applications > Balena1' and a search bar. Below the header, there's a table with columns: Status, Name, Last Seen, UUID, OS Version, OS Variant, Supervisor Version, IP Address, and Release. A single device is listed: 'autumn-water' with status 'Online', last seen 'Currently online (for an hour)', UUID '154cb46', OS Version 'balenaOS 2.36.0+rev2', Supervisor Version '9.15.0', IP Address '10.199.168.237', and Release 'Factory build'. There are also buttons for 'Add filter', 'Add device', 'git remote add balen', 'Views', 'Group actions', and 'Tags'.

Status	Name	Last Seen	UUID	OS Version	OS Variant	Supervisor Version	IP Address	Release
Online	autumn-water	Currently online (for an hour)	154cb46	balenaOS 2.36.0+rev2	balenaOS	9.15.0	10.199.168.237	Factory build

B. Inicialització a Docker de Git vinculat a BalenaOS del nostre device.

1. A la pàgina de l'Aplicació de RESIN.IO, Copiem la ruta que defineix el projecte que hem creat per vincular-lo a git:

The screenshot shows the same Balena.io interface as before, but with a red circle highlighting the 'git remote add balen' button in the top right corner. The button is located next to the 'Views' button. Below the button, there's a table with columns: Supervisor Version, IP Address, and Release. The data is the same as in the previous screenshot: Supervisor Version '9.15.0', IP Address '10.199.168.237', and Release 'Factory build'.

Supervisor Version	IP Address	Release
9.15.0	10.199.168.237	Factory build

En el cas de demostració, nosaltres hem obtingut:

```
git remote add balena gh_ovccvo@git.balena-cloud.com:gh_ovccvo/balena1.git
```

2. Obrim un terminal de Docker QuickStarter

Creem una carpeta amb el nom del projecte i editem el Dockerfile amb les instruccions que volem.

```
mkdir Projecte1
nano Dockerfile
FROM balenalib/raspberrypi3-python:3
ENV INITSYSTEM on
RUN apt-get update
RUN apt-get install python3
CMD ["python3"]
```

3. Seguim les passes per inicialitzar git i fer un push del codi que volem pujar.

```
git clone gh_ovccvo@git.balena-cloud.com:gh_ovccvo/balena1.git
git init
git add .
git remote add balena gh_ovccvo@git.balena-cloud.com:gh_ovccvo/balena1.git
git commit
comentem el fitxer a nano i salvem.
git push balena master
```

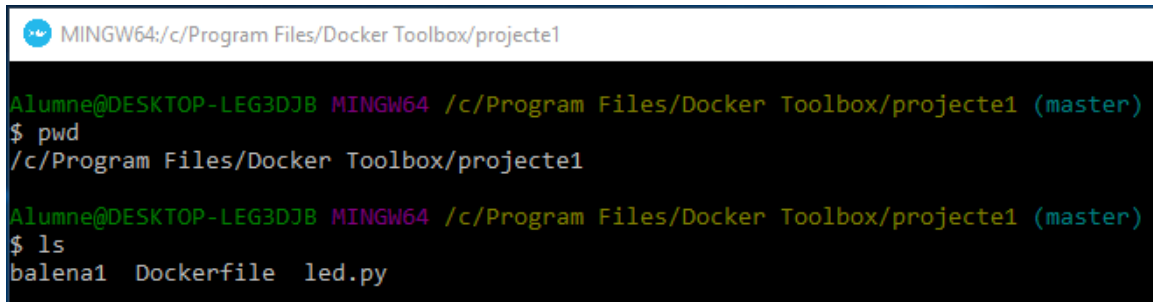
The screenshot displays the Balena Cloud interface for an application named 'autumn-water'. The 'Status' tab is highlighted with a red circle, indicating the application is 'Running'. The 'Logs' tab shows the application's startup logs, including the command 'python3' and an error message 'can't open file "/>

C. Com enviar un codi Python per que s'executi directament a la Raspberry només iniciar el contenidor

1. Creem el codi Python al directori on hem creat el nostre Dockerfile. En el nostre cas:

`/c/Program Files/Docker Toolbox/projecte1`

Hem afegit a la carpeta, el nostre codi al fitxer led.py:



```
MINGW64:/c/Program Files/Docker Toolbox/projecte1

Alumne@DESKTOP-LEG3DJB MINGW64 /c/Program Files/Docker Toolbox/projecte1 (master)
$ pwd
/c/Program Files/Docker Toolbox/projecte1

Alumne@DESKTOP-LEG3DJB MINGW64 /c/Program Files/Docker Toolbox/projecte1 (master)
$ ls
balena1 Dockerfile led.py
```

2. A continuació, editem i afegim tot allò que necessitem per que el nostre codi s'executi dins del Dockerfile que ja teníem:

```
FROM balenalib/raspberrypi3-python:3
ENV INITSYSTEM on
RUN mkdir -p /usr/src/projectebalena1 #Creem el dir projectebalena1 a la ruta
usr/src/
WORKDIR /usr/src/projectebalena1 #Fara que no hagim d'indicar rutes absolutes
quan cridem arxius del directori.
COPY ./led.py /usr/src/projectebalena1
```

```
RUN apt-get update && apt-get install\
    python3 \
    git -y
RUN apt update && apt install -y\
    rpi.gpio \
    curl
```

```
CMD ["python3", "./led.py"] #Afegim l'ordre que es llençarà quan arranqui el
contenidor, com que hem definit el nostre directori de treball, no cal indicar la
ruta absoluta.
```

3. Afegim l'arxiu al GIT per enviar-ho a RESIN.IO

git commit

comentem el fitxer a nano i salvem.

git push balena master

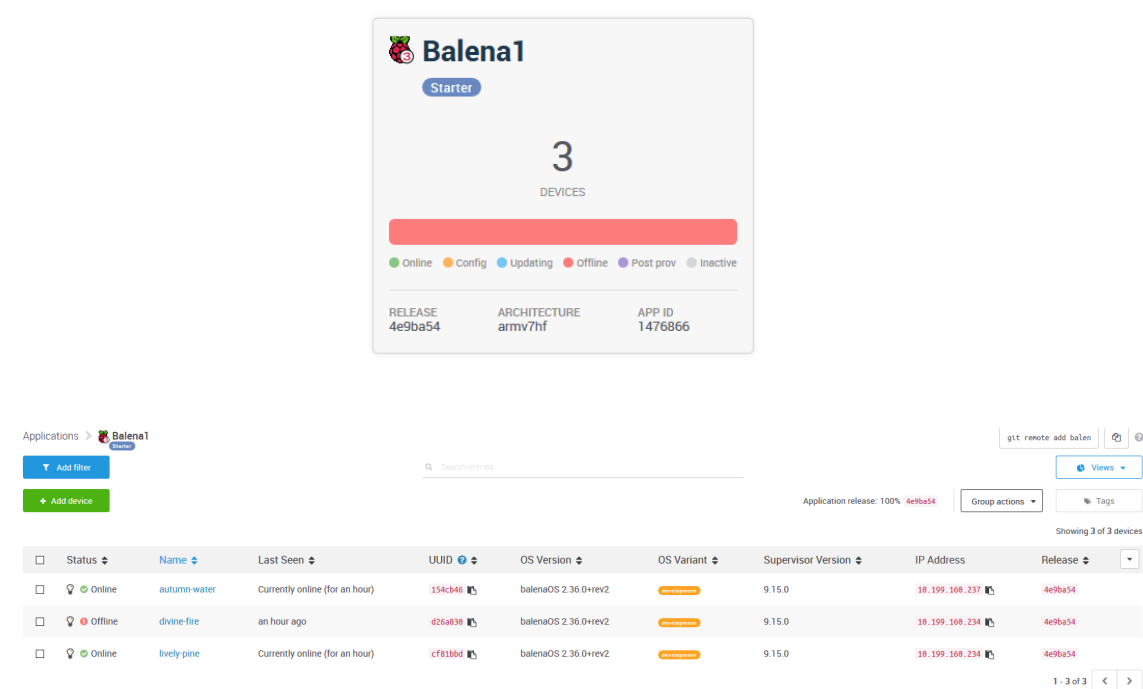
The image displays two screenshots of the BalenaCloud web interface, showing the status of a device named 'autumn-water'.

Top Screenshot: The device is in the 'Installing' state. The 'Status' tab is selected, and the 'main' service is shown as 'Installing'. The 'Logs' panel on the right shows the installation progress, including messages like 'Service exited', 'Killed service', 'Deleting image', and 'Installing service'.

Bottom Screenshot: The device is in the 'Downloaded' state. The 'Status' tab is selected, and the 'main' service is shown as 'Downloaded'. The 'Logs' panel on the right shows the download progress, including messages like 'Downloading image', 'Downloaded image', and 'Killing service'.

D. Afegir nous dispositius a la nostra aplicació

Podem afegir nous devices a una aplicació donada i per tal de fer-ho només ens caldrà cremar la mateixa imatge que ens havíem descarregat desde Resin.io (Balena). Un cop configurada l'adreça IP del nou dispositiu, en arrencar, s'inicialitza de nou per connectarse a Balena Cloud, i ens apareixerà al dashboard de la nostra aplicació:



The image shows the Balena1 Starter dashboard. At the top, it says 'Balena1 Starter' and '3 DEVICES'. Below this is a progress bar and a legend for device status: Online (green), Config (orange), Updating (blue), Offline (red), Post prov (purple), and inactive (grey). Below the legend, it shows 'RELEASE 4e9ba54', 'ARCHITECTURE armv7hf', and 'APP ID 1476866'.

Below the dashboard, there is a table of devices. The table has columns: Status, Name, Last Seen, UUID, OS Version, OS Variant, Supervisor Version, IP Address, and Release. There are three devices listed: 'autumn-water' (Online), 'divine-fire' (Offline), and 'lively-pine' (Online).

Status	Name	Last Seen	UUID	OS Version	OS Variant	Supervisor Version	IP Address	Release
Online	autumn-water	Currently online (for an hour)	154cb46	balenaOS 2.36.0+rev2	balenaOS	9.15.0	18.199.168.237	4e9ba54
Offline	divine-fire	an hour ago	d26a838	balenaOS 2.36.0+rev2	balenaOS	9.15.0	18.199.168.234	4e9ba54
Online	lively-pine	Currently online (for an hour)	cf81b8d	balenaOS 2.36.0+rev2	balenaOS	9.15.0	18.199.168.234	4e9ba54

Automàticament el nou device descarregarà i instal·larà el contenidor associat a la nostra aplicació, de forma que executarà els serveis o aplicacions que tinguem definides.

Adicionalment, a partir d'aquest moment, totes les ordres push que realitzem desde Docker s'enviaran a tots els dispositius que tinguem associats i connectats.

6. Bibliografia utilitzada

1. Best practices for writing Dockerfiles. https://docs.docker.com/develop/develop-images/dockerfile_best-practices/
2. Learn Balena. <https://www.balena.io/docs/learn/welcome/introduction/>
3. Forums balena.io <https://forums.balena.io/>
4. Docker tool box for windows <https://software.intel.com/en-us/intel-system-studio-docker-install-windows-install-docker-toolbox>