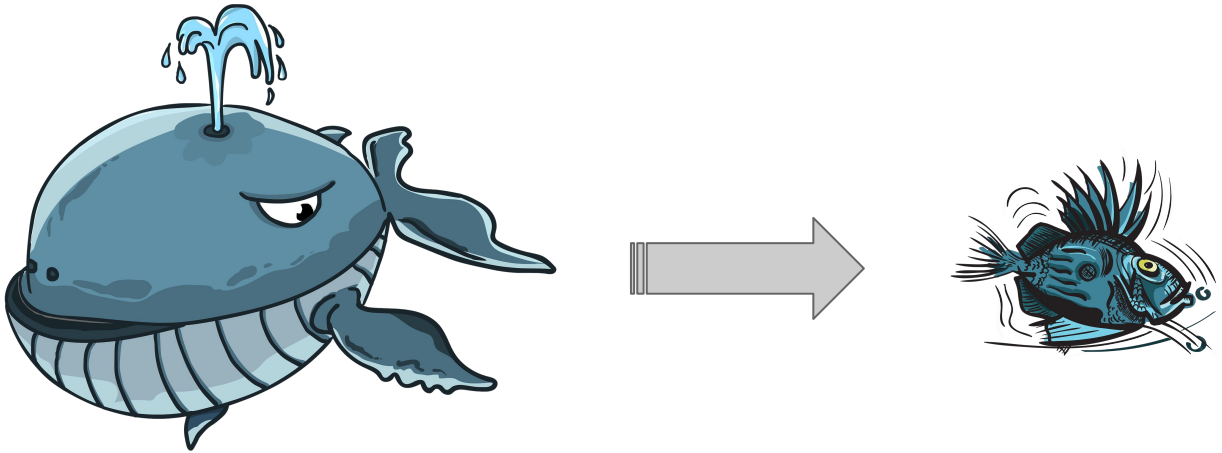
De Ballena a Gallopedro

Optimizando Dockerfiles



esLibre, Granada, Junio 2019

Hola!

Me llamo Ernesto Serrano

- Rompiendo cosas desde 1983
- Desarrollador del *Stevie Wonder Simulator* para Android
- *DevOps Engineer* en **OpenExO**

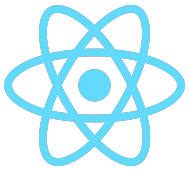
1.

El advenimiento del ExO Monster

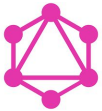
O la dockerización más estúpida de la historia

Arquitectura previa

FRONTEND


API




BACKEND

STORAGE



Tamaño: **8,5Gb**
Tiempo: **20 min**

2.

Hola Lenna!

Ejemplo de código en python para desenfocar a Lenna

```
#!/usr/bin/env python

# This code blurr the "Lenna.png" image file

import os
from PIL import Image, ImageFilter

# Pillow part
in_file = "Lenna.png"
out_file = "Lenna-blurred.png"

original = Image.open(in_file)
blurred = original.filter(ImageFilter.BLUR)
blurred.save(out_file)

print("Original image: %d" % os.stat(in_file).st_size)
print("Blurred image: %d" % os.stat(out_file).st_size)
```

3.

Dockerizemos hermanos

Diferentes formas de dockerizar nuestra aplicación

```
FROM debian:stretch AS example

COPY . .

RUN apt-get update && \
    apt-get install -y \
    python3 \
    python3-pip

RUN pip3 install -r requirements.txt

RUN python3 test.py
```

Tamaño: **515MB**
Tiempo: **57s**

```
FROM debian:stretch-slim AS example

RUN apt-get update && \
    apt-get install -y \
    python3 \
    python3-pip

COPY requirements.txt .

RUN pip3 install -r requirements.txt

COPY . .

RUN python3 test.py
```

Tamaño: **427MB**
Tiempo: **51s**

```
FROM python:3.6-alpine AS example

RUN apk add --no-cache \
    build-base \
    zlib-dev \
    jpeg-dev \
    libpng-dev

COPY requirements.txt .

RUN pip3 install -r requirements.txt

COPY . .

RUN python3 test.py
```

Tamaño: **287MB**
Tiempo: **40s**

```
FROM python:3.6-alpine AS example

COPY requirements.txt .

RUN apk add --no-cache jpeg libpng \
    && \
    apk add --no-cache --virtual .build-deps \
        build-base \
        zlib-dev \
        jpeg-dev \
        libpng-dev \
    && \
        pip3 install -r requirements.txt \
    && \
        apk del .build-deps --force-broken-world

COPY . .

RUN python3 test.py
```

Tamaño: **123MB**
Tiempo: **34s**

```
FROM alpine:3.9 AS example

RUN apk add --no-cache \
    python3 \
    py3-pillow

COPY requirements.txt .

RUN pip3 install -r requirements.txt

COPY . .

RUN python3 test.py
```

Tamaño: **64.9MB**
Tiempo: **6s**

2.

Dockerrizando el rizo

Extrae lo importante y descarta el resto

```
FROM alpine:3.9 AS example

RUN apk add --no-cache python3 py3-pillow

COPY requirements.txt .

RUN pip3 install -r requirements.txt

COPY . .

RUN python3 test.py

FROM scratch AS result

COPY --from=example Lenna-blurred.png .
```

Tamaño: **310kB**
Tiempo: **26ms**

Resultados

REPOSITORY	TAG	SIZE
[secure]/dockerfile-optimization-examples	scratch	310kB
[secure]/dockerfile-optimization-examples	alpine	64.9MB
[secure]/dockerfile-optimization-examples	pythonoptimized	123MB
[secure]/dockerfile-optimization-examples	python	287MB
[secure]/dockerfile-optimization-examples	debianslim	427MB
[secure]/dockerfile-optimization-examples	debian	515MB

debian	stretch-slim	55.3MB
debian	stretch	101MB
python	3.6-alpine	79.1MB
alpine	3.9	5.53MB

Conclusiones

- Utiliza Alpine Linux (**apk add --no-cache**)
- Concatena para reducir el número de capas (**&& **)
- Aprovecha la caché entre capas (**COPY requirements.txt**)
- Extrae solo lo importante (**COPY --from**)

Gracias!

¿Alguna pregunta?

Código disponible en:
https://github.com/erseco/dockerfile-optimization-examples