

# Creación de contenedores propios



## Index

Contenedor con ssh.....	3
Web a través de un ubuntu con nginx.....	6
Entorno de desarrollo con Python.....	7
Docker getting started.....	9
.....	10
Alpine con web bien chula.....	11

# Contenedor con ssh

Vamos a crear un contenedor ssh que ejecute el ssh a través de un debian mismamente, creamos una carpeta con un archivo Dockerfile

```
mkdir practicassh && cd practicassh && nano Dockerfile
```

Le metemos lo siguiente al dockerfile (*modificarlo al gusto del usuario*):

```
FROM debian:latest
```

```
RUN apt-get update && apt-get install -y openssh-server
```

```
RUN mkdir /var/run/sshd
```

```
RUN echo 'root:contraseñaRoot' | chpasswd
```

```
RUN sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config
```

```
EXPOSE 22
```

```
CMD ["/usr/sbin/sshd", "-D"]
```

Alternativamente, si no se quiere que sea el root se puede hacer con un useradd sin problemas:

```
FROM debian:latest
```

```
RUN apt-get update && apt-get install -y openssh-server
```

```
RUN mkdir /var/run/sshd
```

```
RUN useradd -m usuario && echo 'usuario:contraseñaUser' | chpasswd
```

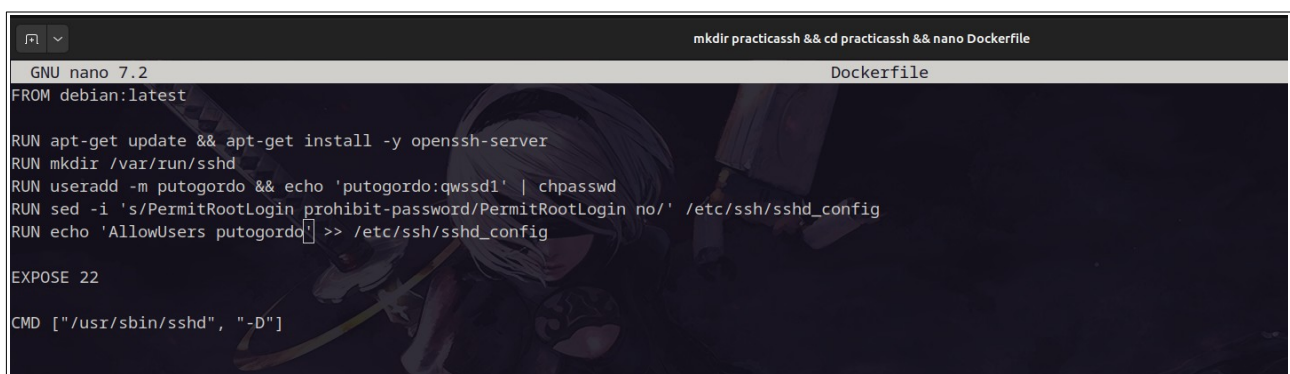
```
RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin no/' /etc/ssh/sshd_config
```

```
RUN echo 'AllowUsers usuario' >> /etc/ssh/sshd_config
```

```
EXPOSE 22
```

```
CMD ["/usr/sbin/sshd", "-D"]
```

Yo en mi caso lo modificare un poco :3

A screenshot of a terminal window with a dark background and a light-colored title bar. The title bar contains the text 'mkdir practicassh && cd practicassh && nano Dockerfile'. The terminal shows the content of the Dockerfile being edited in nano 7.2. The text is as follows:

```
GNU nano 7.2 Dockerfile
FROM debian:latest

RUN apt-get update && apt-get install -y openssh-server
RUN mkdir /var/run/sshd
RUN useradd -m putogordo && echo 'putogordo:qwssd1' | chpasswd
RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin no/' /etc/ssh/sshd_config
RUN echo 'AllowUsers putogordo' >> /etc/ssh/sshd_config

EXPOSE 22

CMD ["/usr/sbin/sshd", "-D"]
```

Y cuando lo tengamos hacer un docker build dentro de la carpeta donde esta el dockerfile, en mi caso, añadiré una cosita:

docker build -t nombreDeLaImagen .

No hace falta el meter **-t nombreDeLaImagen**, pero a mi me gusta meterlo para poder ubicar las imagenes que creamos :3

```
~/Downloads/practicassh on v25.0.2 ...
+ docker build -t babu_ssh .
[+] Building 14.6s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 389B
=> [internal] load metadata for docker.io/library/debian:latest
=> [auth] library/debian:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/6] FROM docker.io/library/debian:latest@sha256:4abf773f2a570e6873259c4e3ba16de6c6268fb571fd46ec80be7c67822823b3
=> => resolve docker.io/library/debian:latest@sha256:4abf773f2a570e6873259c4e3ba16de6c6268fb571fd46ec80be7c67822823b3
=> => sha256:4abf773f2a570e6873259c4e3ba16de6c6268fb571fd46ec80be7c67822823b3 8.52kB / 8.52kB
=> => sha256:f6008b2d1c096556ec301a0c76fe0b40657594ffcfca43aaf63ca9d40767f9a63 1.02kB / 1.02kB
=> => sha256:18f9bd665a29a57601ba643beeb9471f549e3ccff439252551726cddece233a 453B / 453B
=> => sha256:a492eee5e55976c7d3feecce4c564aaf6f14fb07fdc5019d06f4154eddc93fde 48.48MB / 48.48MB
=> => extracting sha256:a492eee5e55976c7d3feecce4c564aaf6f14fb07fdc5019d06f4154eddc93fde
=> [2/6] RUN apt-get update && apt-get install -y openssh-server
=> [3/6] RUN mkdir /var/run/ssh
=> [4/6] RUN useradd -m putogordo && echo 'putogordo:qwssd1' | chpasswd
=> [5/6] RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin no/' /etc/ssh/ssh_config
=> [6/6] RUN echo 'AllowUsers putogordo' >> /etc/ssh/ssh_config
=> exporting to image
=> => exporting layers
=> => writing image sha256:b6bf199c226257bca2568bf8faa2a3c8fb43a70a478c3f14dae191541c520bce
=> => naming to docker.io/library/babu_ssh

What's Next?
View a summary of the build process in the Docker Desktop UI.

~/Downloads/practicassh on v25.0.2 ...
+ docker images
REPOSITORY          TAG          IMAGE ID       CREATED        SIZE
babu_ssh             latest       b6bf199c2262  3 minutes ago  186MB
vsc-react-d9c1fc6154d89dabc4d3c3a399c690a6a2d88239e5363456899d5405a6792dc6-uid  latest       9f09da4bbd4f  2 months ago  1.7GB
vsc-tareas-d8c8b6cb1ccfdcc13759a8c73ea13b1d6fecc36cba48dff8b1422c05f9f87231d-uid  latest       9f09da4bbd4f  2 months ago  1.7GB
welcomepenyas-app   latest       1421ce232115  2 months ago  941MB
phpmyadmin           latest       423af937a83b  7 months ago  562MB
nginx                alpine       099a2d701db1  8 months ago  43.2MB
```

Y una vez pillada la imagen, hacemos un run utilizando el puerto 22:

docker run -d -p 22:22 --name nombreDelContainer nombreDeLaImagen

```
~/Downloads/practicassh on v25.0.2 ...
+ docker run -d -p 22:22 --name babusshizado_container babu_ssh
08a7fe9a54db0d278562927401f918b440abf944c31077dcc16d5d6fffb4a962

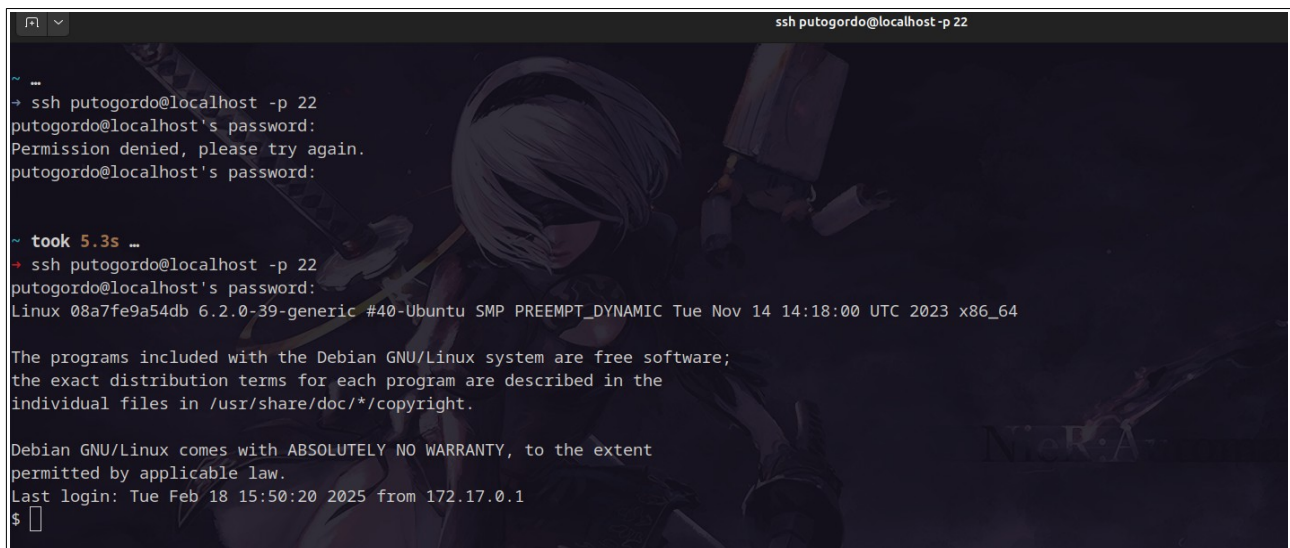
~/Downloads/practicassh on v25.0.2 ...
+ dops
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES
08a7fe9a54db   babu_ssh  "/usr/sbin/sshd -D"      3 seconds ago Up 2 seconds  0.0.0.0:22->22/tcp, :::22->22/tcp  babusshizado_container

~/Downloads/practicassh on v25.0.2 ...
+ █
```

Y ahora si ya queremos, con el comando:

```
ssh usuarioDelContainerQueCreamosAntes@localhost -p 22
```

Podemos usar ssh como queramos:



```
ssh putogordo@localhost -p 22
putogordo@localhost's password:
Permission denied, please try again.
putogordo@localhost's password:

~ took 5.3s ...
+ ssh putogordo@localhost -p 22
putogordo@localhost's password:
Linux 08a7fe9a54db 6.2.0-39-generic #40-Ubuntu SMP PREEMPT_DYNAMIC Tue Nov 14 14:18:00 UTC 2023 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Feb 18 15:50:20 2025 from 172.17.0.1
$
```

# Web a través de un ubuntu con nginx

Siguiendo el esquema que hicimos anteriormente, creare un Dockerfile con el siguiente contenido:

```
FROM nginx:latest
```

```
EXPOSE 80
```

```
CMD ["nginx", "-g", "daemon off;"]
```

Ejecutaremos un contenedor de nginx y finiquitado.

```
AWS_apuntes/No_AWS/February/Task5.5_PersonalContainers/webDockerizada on main [?] on v25.0.2 ...
→ docker build -t mr-web-joselito .
[+] Building 1.4s (6/6) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 101B
=> [internal] load metadata for docker.io/library/nginx:latest
=> [auth] library/nginx:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> CACHED [1/1] FROM docker.io/library/nginx:latest@sha256:91734281c0ebfc6f1aea979cffe5079cfe786228a71cc6f1f46a228c
=> exporting to image
=> => exporting layers
=> => writing image sha256:5d59f8097693a91fdec14a4e778715379fd9373a6ee842a3086b2f5ea1cd80b3
=> => naming to docker.io/library/mr-web-joselito

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview

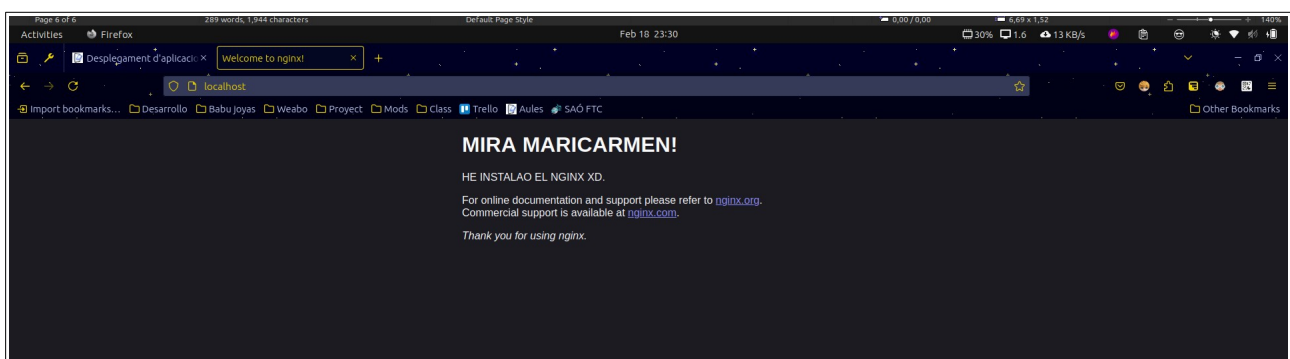
AWS_apuntes/No_AWS/February/Task5.5_PersonalContainers/webDockerizada on main [?] on v25.0.2 ...
→
```

```
docker run -d -p 80:80 --name nombreDelContainer nombreDeLaImagen
```

```
AWS_apuntes/No_AWS/February/Task5.5_PersonalContainers/webDockerizada on main [?] on v25.0.2 ...
→ docker run -d -p 80:80 --name WebSencillaJoselito mr-web-joselito
5b3b251235233428dd03f5c3e6157215309d27575e85fec3ac20b63b04c9fa5e selected (64 bytes)

AWS_apuntes/No_AWS/February/Task5.5_PersonalContainers/webDockerizada on main [?] on v25.0.2 ...
→ dops
CONTAINER ID    IMAGE           COMMAND          CREATED        STATUS        PORTS                               NAMES
5b3b25123523    mr-web-joselito  "/docker-entrypoint..."  2 seconds ago  Up 2 seconds  0.0.0.0:80->80/tcp, :::80->80/tcp  WebSencillaJoselito

AWS_apuntes/No_AWS/February/Task5.5_PersonalContainers/webDockerizada on main [?] on v25.0.2 ...
→
```





# Entorno de desarrollo con Python

Acá al igual que en el primer paso vamos a crear un contenedor para poder trabajar con python a través de un contenedor.

Conenido del Dockerfile:

FROM python:3

RUN mkdir -p /pythonPruebas/gord

WORKDIR /pythonPruebas/gord

CMD [ "python" ]

```
AWS_apuntes/No_AWS/February/Task5.5_PersonalContainers/Pythoneador on main [?] on v25.0.2 ...
$ docker build -t pitoneador .
[+] Building 1.9s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 130B
=> [internal] load metadata for docker.io/library/python:3
=> [auth] library/python:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/3] FROM docker.io/library/python:3@sha256:08471c63c5fdf2644adc142a7fa8d0290eb405cda14c473f5e5b4cd0933af601
=> [2/3] RUN mkdir -p /pythonPruebas/gord
=> [3/3] WORKDIR /pythonPruebas/gord
=> exporting to image
=> => exporting layers
=> => writing image sha256:1c0d1851led105efc4173e1fd0d6f94eaf33bf253736dd08539b402d7ffefddb
=> => naming to docker.io/library/pitoneador

What's Next?
View a summary of image vulnerabilities and recommendations - docker scout quickview

AWS_apuntes/No_AWS/February/Task5.5_PersonalContainers/Pythoneador on main [?] on v25.0.2 took 2.0s ...
```

Y el resto como antes y hacer el docker run:

docker run -d -it --name nombreContainer nombreImagen

Aviso, tal cuál lo he hecho es una imagen sin mucho sentido, de normal, se busca pillar archivos del sistema, y hacer que cuando enciendas o accedas al repositorio se active el archivo de python.

Acá solo he definido donde se guardarían esos archivos:

FROM python:3

WORKDIR /usr/src/app

COPY requirements.txt ./

RUN pip install --no-cache-dir -r requirements.txt

# En esta línea de acá si quisieses, puedes meter un archivo python para ejecutarlo. Si no no pongas  
# esta línea de abajo.

COPY . .

CMD [ "python", "./your-daemon-or-script.py" ]

En este ejemplo que se ve acá se hace instalación de dependencias con requirements.txt.

El copy vacío es para copiar los archivos que te interesen al workdir.

```
joselito@kyogre:~/Documents/Tareas/AWS_apuntes/No_AWS/February/Task5.5_PersonalContainers/Pythoneador
# exit

AWS_apuntes/No_AWS/February/Task5.5_PersonalContainers/Pythoneador on main [?] on v25.0.2 took 25.1s ...
→ dops
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS          PORTS          NAMES
defc12dd51a0   pitoneador "python"                About a minute ago Up About a minute          PythonGordo

AWS_apuntes/No_AWS/February/Task5.5_PersonalContainers/Pythoneador on main [?] on v25.0.2 ...
+ █

docker exec -it defc12dd51a0 bash

+ docker exec -it defc12dd51a0 bash
root@defc12dd51a0:/pythonPruebas/gord# █
```



# Docker getting started

Pillaremos de github el repositorio ya creado:

```
git clone https://github.com/docker/getting-started-app.git
```

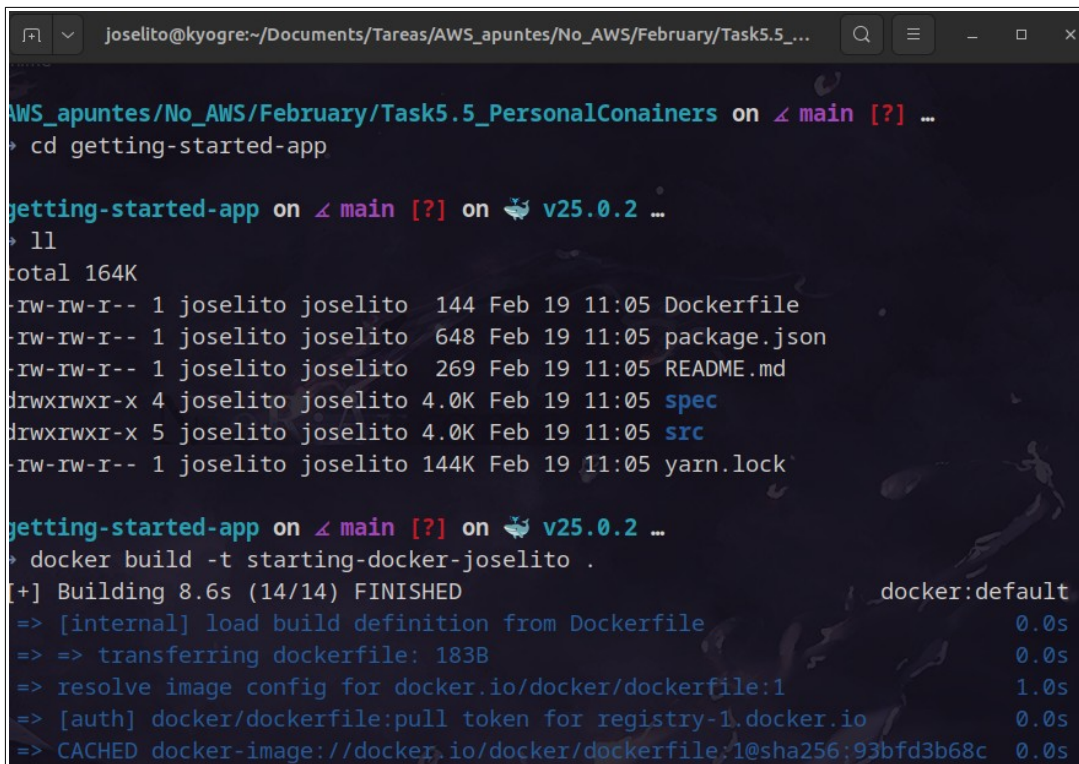
Una vez pillado vamos dentro del repositorio recién creado y crearemos un dockerfile, le meteremos mismamente el node alpine del ejemplo:

```
# syntax=docker/dockerfile:1
```

```
FROM node:lts-alpine
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000
```

Y lo ejecutamos:

```
docker build -t tuto-de-docker . && docker run -d -p 127.0.0.1:3000:3000 --name TutoJoselito starting-docker-joselito
```

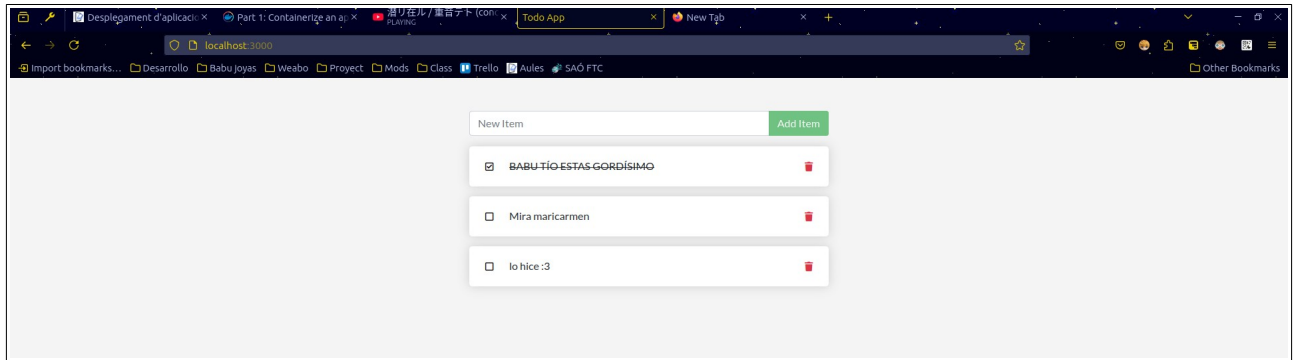


```
joshito@kyogre:~/Documents/Tareas/AWS_apuntes/No_AWS/February/Task5.5_...
AWS_apuntes/No_AWS/February/Task5.5_PersonalContainers on  main [?] ...
> cd getting-started-app

getting-started-app on  main [?] on  v25.0.2 ...
> ll
total 164K
-rw-rw-r-- 1 joshito joshito 144 Feb 19 11:05 Dockerfile
-rw-rw-r-- 1 joshito joshito 648 Feb 19 11:05 package.json
-rw-rw-r-- 1 joshito joshito 269 Feb 19 11:05 README.md
drwxrwxr-x 4 joshito joshito 4.0K Feb 19 11:05 spec
drwxrwxr-x 5 joshito joshito 4.0K Feb 19 11:05 src
-rw-rw-r-- 1 joshito joshito 144K Feb 19 11:05 yarn.lock

getting-started-app on  main [?] on  v25.0.2 ...
> docker build -t starting-docker-joselito .
[+] Building 8.6s (14/14) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 183B                               0.0s
=> resolve image config for docker.io/docker/dockerfile:1        1.0s
=> [auth] docker/dockerfile:pull token for registry-1.docker.io  0.0s
=> CACHED docker-image://docker.io/docker/dockerfile:1@sha256:93bfd3b68c 0.0s
```

Ahora si vamos a <http://localhost:3000>, tendremos del getting started un todo list bien cute :3



# Alpine con web bien chula

Crearemos un dockerfile de alpine en el que pillaremos y meteremos php y nginx. Y le meteremos un phpinfo doc para mostrar q el php funcione perf:

FROM alpine:3.14

RUN apk add --no-cache nginx php php7-fpm

RUN echo "<p>Holi :3</p><br><?php phpinfo(); ?>" > /var/www/localhost/htdocs/index.php

RUN mkdir -p /run/nginx && \

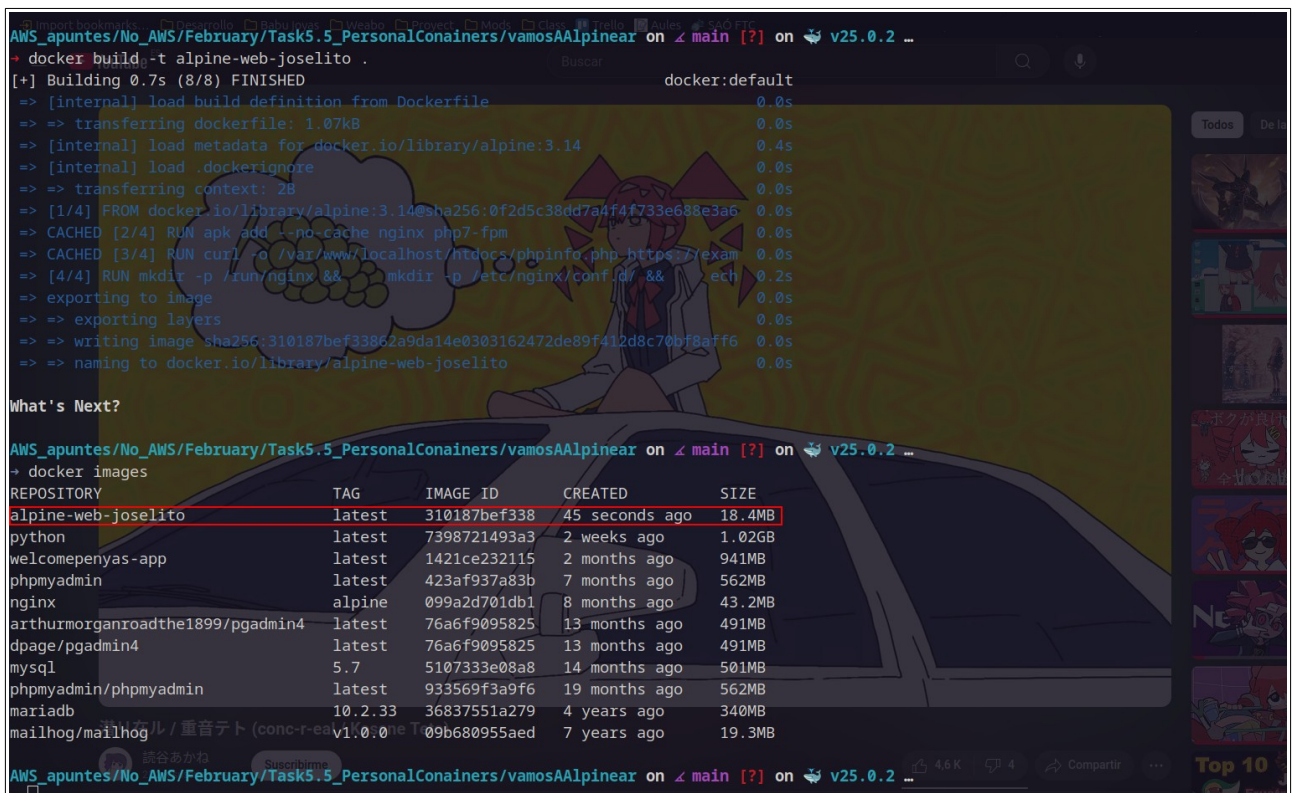
mkdir -p /etc/nginx/conf.d/ && \

echo 'server { listen 80; server\_name localhost; root /var/www/localhost/htdocs; index index.php index.html; location / { try\_files \$uri \$uri/ /index.php; } }' > /etc/nginx/conf.d/default.conf

EXPOSE 80

CMD php-fpm7 && nginx -g 'daemon off;'

Y ahorita como siempre hacemos un docker build :3

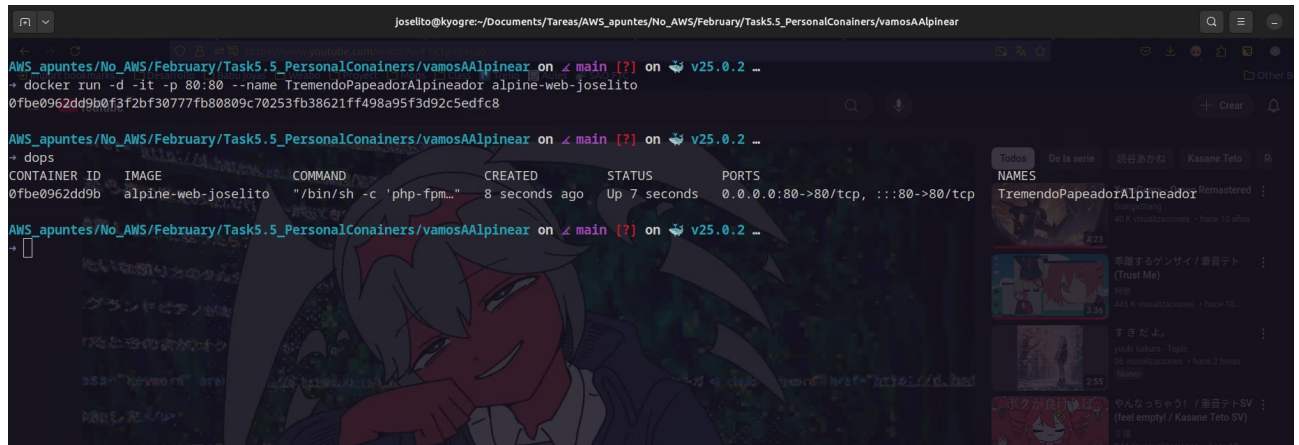


```
AWS_apuntes/No_AWS/February/Task5.5_PersonalContainers/vamosAAlpinear on main [?] on v25.0.2 ...
→ docker build -t alpine-web-joselito .
[+] Building 0.7s (8/8) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 1.07kB 0.0s
=> [internal] load metadata for docker.io/library/alpine:3.14 0.4s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/4] FROM docker.io/library/alpine:3.14@sha256:0f2d5c38dd7a4f4f733e688e3a6 0.0s
=> CACHED [2/4] RUN apk add --no-cache nginx php7-fpm 0.0s
=> CACHED [3/4] RUN curl -o /var/www/localhost/htdocs/phpinfo.php https://exam 0.0s
=> [4/4] RUN mkdir -p /run/nginx && mkdir -p /etc/nginx/conf.d/ && ech 0.2s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:310187bef33862a9da14e0303162472de89f412d8c70bf8aff6 0.0s
=> => naming to docker.io/library/alpine-web-joselito 0.0s

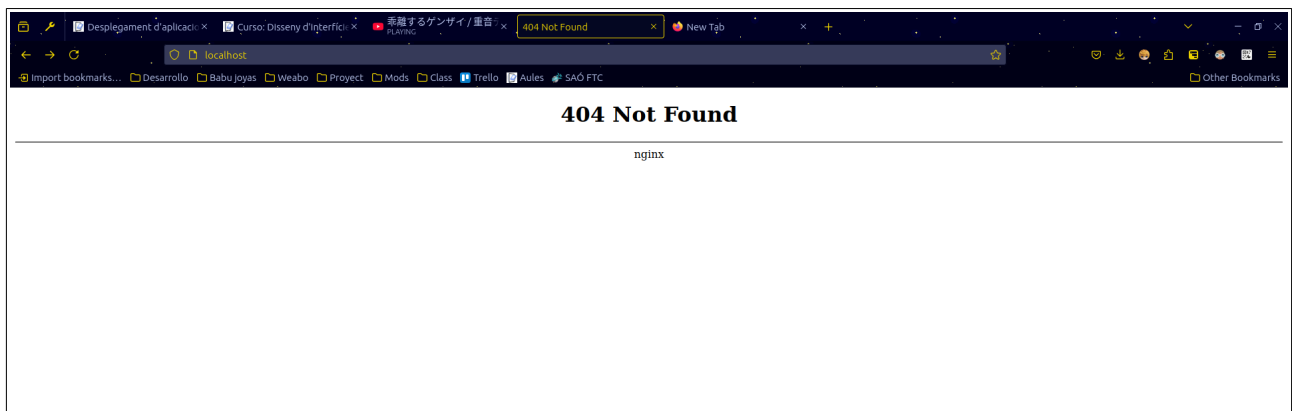
What's Next?
AWS_apuntes/No_AWS/February/Task5.5_PersonalContainers/vamosAAlpinear on main [?] on v25.0.2 ...
→ docker images
REPOSITORY          TAG          IMAGE ID      CREATED      SIZE
alpine-web-joselito latest       310187bef338 45 seconds ago 18.4MB
python              latest       7398721493a3 2 weeks ago  1.02GB
welcomepenyas-app  latest       1421ce232115 2 months ago  941MB
phpmyadmin          latest       423af937a83b 7 months ago  562MB
nginx               alpine      099a2d701db1 8 months ago  43.2MB
arthurmorganroadthe1899/pgadmin4 latest       76a6f9095825 13 months ago 491MB
dpag/pgadmin4       latest       76a6f9095825 13 months ago 491MB
mysql               5.7         5107333e08a8 14 months ago 501MB
phpmyadmin/phpmyadmin latest       933569f3a9f6 19 months ago 562MB
mariadb             10.2.33     36837551a279 4 years ago  340MB
mailhog/mailhog     v1.0.0      09b68095aed 7 years ago  19.3MB
```

Y ahora el run y arreglao, accederemos a <http://localhost> y deberiamos ver bien bonito una web de prueba en un index.php con info de php que hemos insertado :3

docker run -d -it -p 80:80 --name TremendoPapeadorAlpineador alpine-web-joselito



Se abre el container pero en mi caso me da error 404 de que no puede acceder al index.php en /var/www/localhost/htdocs/



### Nota para el profesor:

*Me he pasado un ratico mirando, pero no logré encontrar el error, sriry Maricarmen, pero no me da la vida con todo slos trabajos, literal me pase como 2 horicas mirando como bobo haciendolo a prisas, lo más probable esque me haya obviado una tontería monumental como que el default.conf lo he de meter en otro sitio, o haya de meter un archivo de nginx que se me ha olvidado.*

*Cuando tenga un ratete trataré de darle revancha, sriry, no puedo perder más tiempo en esta actividad ^u*