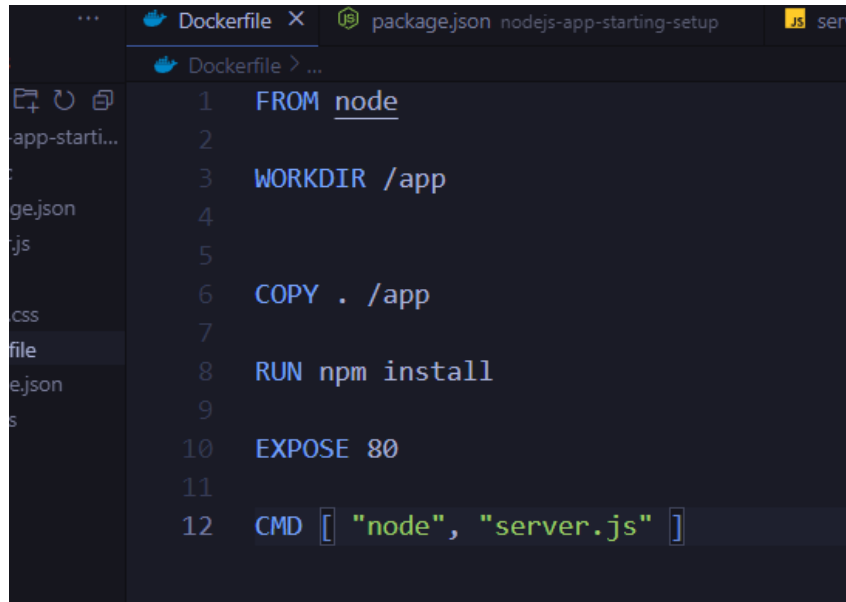


Laboratorio 4A.

Crear un archivo Dockerfile. Operaciones sobre contenedores.

Paso 1. Descargamos la carpeta comprimida con los archivos.

Creamos el **Dockerfile** sin la extensión.

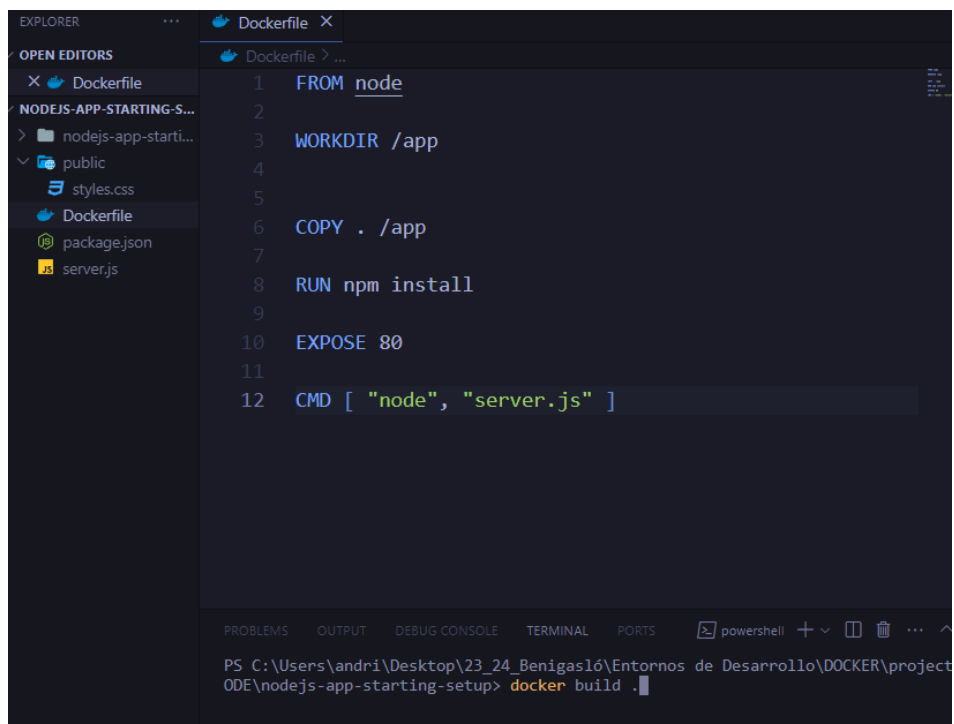


```
1 FROM node
2
3 WORKDIR /app
4
5
6 COPY . /app
7
8 RUN npm install
9
10 EXPOSE 80
11
12 CMD [ "node", "server.js" ]
```

Paso 2. Creamos la imagen sobre el Dockerfile:

Utilizamos el comando ya conocido:

docker build . (dejamos un espacio y colocamos un punto)



```
1 FROM node
2
3 WORKDIR /app
4
5
6 COPY . /app
7
8 RUN npm install
9
10 EXPOSE 80
11
12 CMD [ "node", "server.js" ]
```

```
PS C:\Users\andri\Desktop\23_24_Benigasl6\Entornos de Desarrollo\DOCKER\projectODE\nodejs-app-starting-setup> docker build .
```



Resultado:

```
PS C:\Users\andri\Desktop\23_24_Benigasló\Entornos de Desarrollo\DOCKER\project 2NODE\nodejs-app-starting-setup> docker build .
[+] Building 3.3s (9/9) FINISHED
=> => transferring dockerfile: 142B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/node:latest
=> [1/4] FROM docker.io/library/node
=> [internal] load build context
=> => transferring context: 16.56kB
=> [2/4] WORKDIR /app
=> [3/4] COPY . /app
=> [4/4] RUN npm install
=> exporting to image
=> exporting layers
=> writing image sha256:df7a55bd5f6c612be7d9db22a57fb4171a1f7e0778044f7 0.0s

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

Podemos observar los pasos que hemos especificado en el Dockerfile, se han ejecutado sin errores.

Paso 3. Buscamos el nombre de la imagen y creamos una instancia un proceso, es decir un contenedor. Por lo tanto, copiamos el ID de la imagen y en la terminal de visual estudio code escribimos:

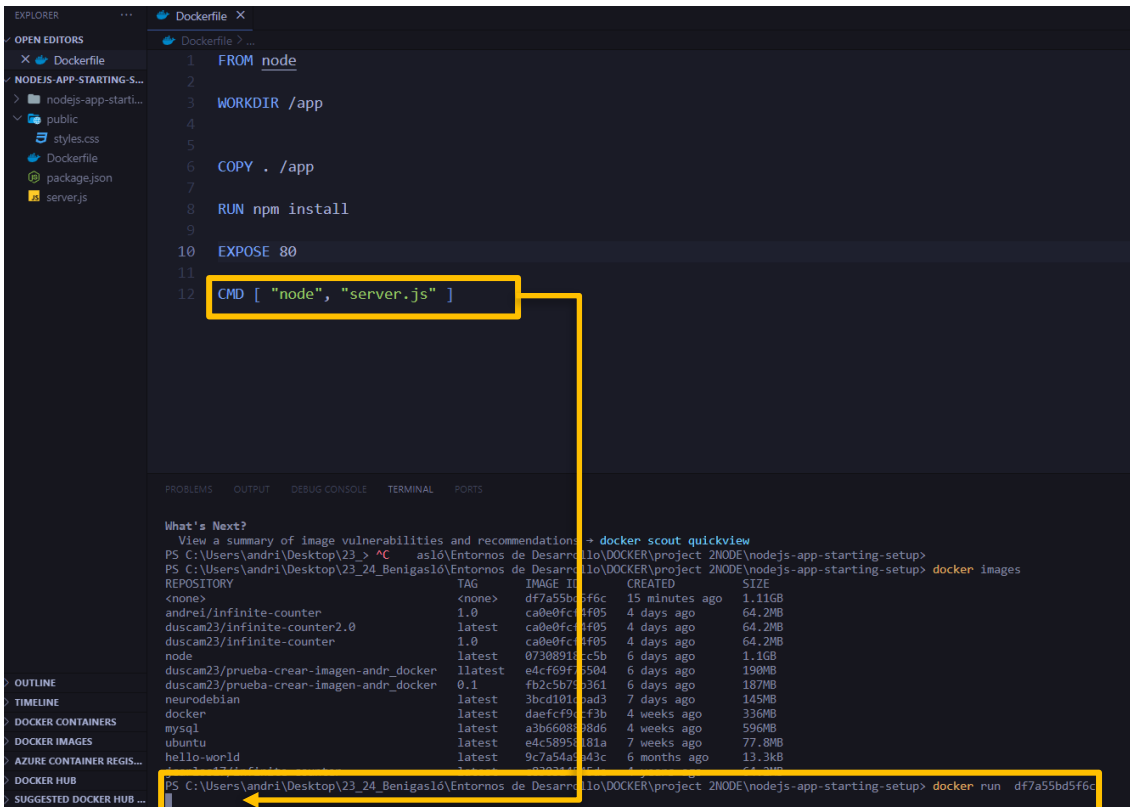
```
jonpitsaf@info2-23:~/Escritorio/Entornos/4A$ docker build .
[+] Building 50.6s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 128B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/node:latest
=> [auth] library/node:pull token for registry-1.docker.io
=> [1/4] FROM docker.io/library/node@sha256:84bb4077fd52933a935e7057ba9991e7cb18487b0ba444835dd44975aa94b7b2
=> resolve docker.io/library/node@sha256:84bb4077fd52933a935e7057ba9991e7cb18487b0ba444835dd44975aa94b7b2
=> sha256:06facfb3aace8ca5f47e2895901e4f1bba25f9b1f401cc009708cc05225e115 2.00kB / 2.00kB
=> sha256:90e5e7d8b7a34877f61c2b86d053db1c4f440b9054cf49573e3be5d6a674a47 49.58MB / 49.58MB
=> sha256:27e1a8ca91d35598fbae8dee7f1c211f0f93cec529f6804a60e9301c53a604d0 24.05MB / 24.05MB
=> sha256:d3a767d1d12e57724b9f254794e359f3b04d4d5ad966006e5b5cda78cc382762 64.13MB / 64.13MB
=> sha256:84bb4077fd52933a935e7057ba9991e7cb18487b0ba444835dd44975aa94b7b2 1.21kB / 1.21kB
=> sha256:07308918cc5b9f838e11922e6626826b380730bc85fa0dff39e84a8683a62966 7.53kB / 7.53kB
=> sha256:711be5dc50448ab08ccab0b44d65962f36574d341749ab30651b78ec0d4bfd1c 211.07MB / 211.07MB
=> sha256:22956530cc64ef2361591684e23e3b8e5bb5910da23197635a2b5b96a34b488d 3.37kB / 3.37kB
=> sha256:f72e9eb5a1b39e6b6969a2b02e57c8be809fac89205a98594f69bf532e74d4db 49.33MB / 49.33MB
=> extracting sha256:90e5e7d8b7a34877f61c2b86d053db1c4f440b9054cf49573e3be5d6a674a47
=> sha256:96d22460c6db052a986434847a4a217967597989687e0bb01faf52208c78207a 2.23MB / 2.23MB
=> extracting sha256:27e1a8ca91d35598fbae8dee7f1c211f0f93cec529f6804a60e9301c53a604d0
=> sha256:4e772c30a8d40e1deba91b6961381e3b931fa4839cc4caf6de13f05ca975e802 451B / 451B
=> extracting sha256:d3a767d1d12e57724b9f254794e359f3b04d4d5ad966006e5b5cda78cc382762
=> extracting sha256:711be5dc50448ab08ccab0b44d65962f36574d341749ab30651b78ec0d4bfd1c
=> extracting sha256:22956530cc64ef2361591684e23e3b8e5bb5910da23197635a2b5b96a34b488d
=> extracting sha256:f72e9eb5a1b39e6b6969a2b02e57c8be809fac89205a98594f69bf532e74d4db
=> extracting sha256:96d22460c6db052a986434847a4a217967597989687e0bb01faf52208c78207a
=> extracting sha256:4e772c30a8d40e1deba91b6961381e3b931fa4839cc4caf6de13f05ca975e802
=> [internal] load build context
=> => transferring context: 2.06kB
=> [2/4] WORKDIR /app
=> [3/4] COPY . /app
=> [4/4] RUN npm install
=> exporting to image
=> exporting layers
=> writing image sha256:df1c8c6a31f5a515179bfa52f4a07f7f65af4c3241b3bca46240203f16da587b
jonpitsaf@info2-23:~/Escritorio/Entornos/4A$
```

docker run df7a

El contenedor no se para. El motivo es que le hemos indicado esta instrucción de start un nuevo servidor:



```
jonpitsaf@info2-23:~/Escritorio/Entornos/4A$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
<none>        <none>    df1c8c6a31f5   6 minutes ago  1.11GB
mysql         latest    a3b6608898d6   5 weeks ago    596MB
ubuntu        latest    e4c58958181a   7 weeks ago    77.8MB
hello-world    latest    9c7a54a9a43c   6 months ago   13.3kB
jonpitsaf@info2-23:~/Escritorio/Entornos/4A$ docker run df1c8c6a31f5
```



Paso 4. Abrimos un otro terminal en la parte derecha de la terminal VSC y configuramos los surgentes pasos .



Paso 5. Para ver todos los procesos activos, en la terminal nueva escribimos el comando `docker ps -a`
 Ahora el siguiente paso es parar nuestro contenedor. Lo haremos con el comando

```
jonpitsaf@info2-23:~/Escritorio/Entornos/4A$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
c9da893cdf2d	df1c8c6a31f5	"docker-entrypoint.s..."	34 seconds ago	Up 34 seconds	80/tcp	determined_mestorf
494d9dba07f3	df1c8c6a31f5	"docker-entrypoint.s..."	3 minutes ago	Up 3 minutes	80/tcp	flamboyant_liskov
305f3bcdff9	df1c8c6a31f5	"docker-entrypoint.s..."	4 minutes ago	Up 4 minutes	80/tcp	vigilant_shirley
6fb7d1c271b8	ubuntu	"bash"	12 days ago	Exited (0) 9 days ago		nostalgic_gould
e9b69100b382	ubuntu	"/bin/bash"	12 days ago	Exited (0) 12 days ago		nervous_nash
a8a5496b42a2	hello-world	"/hello"	12 days ago	Exited (0) 12 days ago		admiring_bell
44b94f156a4b	ubuntu	"/bin/bash"	12 days ago	Exited (0) 12 days ago		great_gould
502a425b59ec	mysql:latest	"docker-entrypoint.s..."	2 weeks ago	Exited (0) 13 days ago		mi-mysql
5c7e45ab87ce	hello-world	"/hello"	2 weeks ago	Exited (0) 2 weeks ago		zen_brattain

`docker stop nombre del contenedor o ID`

```
upbeat_dhawan
PS C:\Users\andri\Desktop\23_24_BenigasId\Entornos de Desarrollo\DOCKER\project 2NODE\nodejs-app-starting-setup> docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
6222c47800f6	df7a55bd5f6c	"docker-entrypoint.s..."	23 minutes ago	Exited (137) 8 seconds ago		upbeat_dhawan
b55a66a5a8c6	node:latest	"docker-entrypoint.s..."	9 hours ago	Exited (11) 8 hours ago		kind_neumann

```
jonpitsaf@info2-23:~/Escritorio/Entornos/4A$ docker stop c9da893cdf2d
c9da893cdf2d
```

Y después veremos el comando **exited** que significa el proceso finalizado.

Paso 6. Si buscamos ver la respuesta en nuestro localhost 80, el resultado nos dará un error. Necesitamos realizar unos pasos opcionales. Volver a ejecutar el contenedor, pero necesitamos agregar una bandera adicional. Es decir:

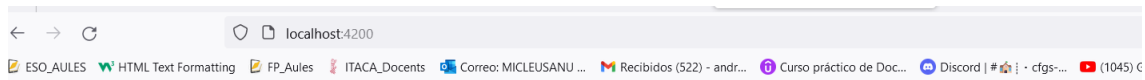
Añadimos la bandera `-p` delante del nombre contenedor.

`docker run -p 4200:80 df7a55bd5f6c`

observamos en localhost 4200 respuesta: `http://localhost:4200/`

```
jonpitsaf@info2-23:~/Escritorio/Entornos/4A$ docker run -p 4200 df1c8c6a31f5
```





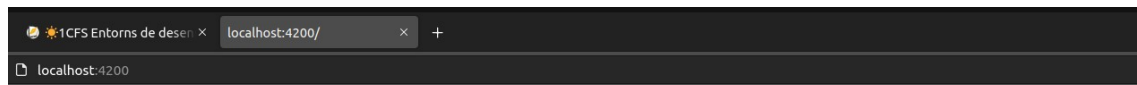
My Course Goal

Learn Docker!

Course Goal

👋 Saludar al 🇪🇸 Juan

Set Course Goal



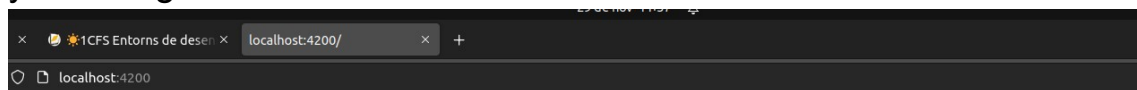
My Course Goal

Learn Docker!

Course Goal

Set Course Goal

Paso 7. Cada alumno debe colocar un mensaje personalizado en la categoría **"My Course Goal"**. En otras palabras, una frase, y también realizar una captura de pantalla demostrando que el código y la configuración están funcionando correctamente.



My Course Goal

Jonas al poder

Course Goal

Set Course Goal

Parte B. Operaciones y gestiones con los contenedores.



Paso B1. Crear Contenedores con Puertos Locales Específicos y Terminal Interactiva:

```terminal

```
docker run -d -it --name contenedor1 -p 6000:puerto_interno
nombre_de_tu_imagen
```

```
jonpitsaf@info2-23:~/Escritorio/Entornos/4AS$ docker run -d -it --name contenedor1 -p 6000:80 df1c8c6a31f5
1192d8ecfbb3662d028542115dff5d44a80843ed22e1322884cdabc9780070dc
jonpitsaf@info2-23:~/Escritorio/Entornos/4AS$
```

Ln 1, Col 1 Sp

```
docker run -d -it --name contenedor2 -p 6500:puerto_interno
nombre_de_tu_imagen
```

```
jonpitsaf@info2-23:~/Escritorio/Entornos/4AS$ docker run -d -it --name contenedor2 -p 6500:80 df1c8c6a31f5
4fd93c10e696f4c406bb25e97a855f15270ee82590402bb2b983e4668b80eebb
jonpitsaf@info2-23:~/Escritorio/Entornos/4AS$
```

Ln 1, Col 1 Sp

```
docker run -d -it --name contenedor3 -p 7000:puerto_interno
nombre_de_tu_imagen
```

```
jonpitsaf@info2-23:~/Escritorio/Entornos/4AS$ docker run -d -it --name contenedor3 -p 7000:80 df1c8c6a31f5
dfdclb5b7653a3c8b91139526056971bbe09debe5d8b7abf3d0c8e259b35c438
jonpitsaf@info2-23:~/Escritorio/Entornos/4AS$
```

Ln 1, Col 1 Sp

```

Asegúrate de reemplazar "puerto_interno" **busca en dockerhub**
<https://hub.docker.com/layers/library/node/lts-slim/images/sha256-18aacc7993a16f1d766c21e3bff922e830bcd7b549bbb789ceb7374a6138480?context=explore>

, o en

<https://es.stackoverflow.com/search?q=puerto+expose+node>

el puerto de la imagen! con el número o de puerto que tu aplicación dentro del contenedor está utilizando.

Paso B2. Cambiar Nombre del Contenedor:

```terminal

```
docker rename contenedor1 nuevo_nombre_contenedor1
```

```
See 'docker --help'
jonpitsaf@info2-23:~/Escritorio/Entornos/4AS$ docker rename contenedor1 Jonas1
jonpitsaf@info2-23:~/Escritorio/Entornos/4AS$
```

Ln 1, Col 1 Sp

```



Paso B3. **Adjuntar un Archivo al Contenedor en Ejecución:**

``` terminal

```
docker cp archivo_adjunto.txt contenedor2:/ruta/destino
```

```

Paso B4. **Detener uno de los Contenedores:**

``` terminal

```
docker stop contenedor3
```

```

```
jonpitsaf@info2-23:~/Escritorio/Entornos/4A$ docker stop contenedor3
contenedor3
jonpitsaf@info2-23:~/Escritorio/Entornos/4A$
```

Paso B5. **Abrir Terminal Interactiva (bash) en uno de los Contenedores:**

``` terminal

```
docker exec -it contenedor1 /bin/bash
```

```
contenedor3
jonpitsaf@info2-23:~/Escritorio/Entornos/4A$ docker exec -it Jonasl /bin/bash
root@1192d8ecfbb3:/app#
```

```

Esto abrirá una terminal interactiva en el contenedor1.

Paso B6. Sugerencia Adicional: Verificar Estadísticas del Contenedor:

``` terminal

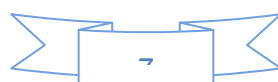
```
docker stats contenedor2
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS
4fd93c10e696 contenedor2 0.00% 0B / 0B 0.00% 0B / 0B 0B / 0B 0
```

```

Este comando muestra estadísticas en tiempo real del uso de recursos del contenedor2.



Paso B7. Ver Historial de la Imagen:

``` terminal

docker history nombre\_de\_tu\_imagen

```
jonpitsaf@info2-23:~/Escritorio/Entornos/4A$ docker history df1c8c6a31f5
IMAGE CREATED CREATED BY SIZE COMMENT
df1c8c6a31f5 46 hours ago CMD ["node" "server.js"] 0B buildkit.dockerfile.v0
<missing> 46 hours ago EXPOSE map[80/tcp:{}] 0B buildkit.dockerfile.v0
<missing> 46 hours ago RUN /bin/sh -c npm install # buildkit 7.07MB buildkit.dockerfile.v0
<missing> 46 hours ago COPY . /app # buildkit 1.87kB buildkit.dockerfile.v0
<missing> 46 hours ago WORKDIR /app 0B buildkit.dockerfile.v0
<missing> 9 days ago /bin/sh -c #(nop) CMD ["node"] 0B
<missing> 9 days ago /bin/sh -c #(nop) ENTRYPOINT ["docker-entry... 0B
<missing> 9 days ago /bin/sh -c #(nop) COPY file:4d192565a7220e13... 388B
<missing> 9 days ago /bin/sh -c set -ex && export GNUPGHOME="$(... 7.58MB
<missing> 9 days ago /bin/sh -c #(nop) ENV YARN_VERSION=1.22.19 0B
<missing> 9 days ago /bin/sh -c ARCH= && dpkgArch="$(dpkg --print... 166MB
<missing> 9 days ago /bin/sh -c #(nop) ENV NODE_VERSION=21.2.0 0B
<missing> 9 days ago /bin/sh -c groupadd --gid 1000 node && use... 8.94kB
<missing> 9 days ago /bin/sh -c set -ex; apt-get update; apt-ge... 587MB
<missing> 9 days ago /bin/sh -c apt-get update && apt-get install... 177MB
<missing> 9 days ago /bin/sh -c set -eux; apt-get update; apt-g... 48.4MB
<missing> 10 days ago /bin/sh -c #(nop) CMD ["bash"] 0B
<missing> 10 days ago /bin/sh -c #(nop) ADD file:39d17d28c5de0bd62... 117MB
```

```

Este comando mostrará el historial de capas de la imagen Docker.

Paso B8. Obtener Detalles del Contenedor mediante `docker inspect`:

``` terminal

docker inspect contenedor1

```

```
jonpitsaf@info2-23:~/Escritorio/Entornos/4A$ docker inspect Jonas1
[
  {
    "Id": "1192d8ecfbb3662d028542115dff5d44a80843ed22e1322884cdabc9780070dc",
    "Created": "2023-11-29T11:04:56.180956043Z",
    "Path": "docker-entrypoint.sh",
    "Args": [
      "node",
      "server.js"
    ],
    "State": {
      "Status": "exited",
      "Running": false,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 0,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2023-11-29T11:04:56.469841761Z",
      "FinishedAt": "2023-11-30T07:07:56.153037312Z"
    },
    "Image": "sha256:df1c8c6a31f5a515179bfa52f4a07f7f65af4c3241b3bca40240203f10da587b",
    "ResolveConfPath": "/home/jonpitsaf/.local/share/docker/containers/1192d8ecfbb3662d028542115dff5d44a80843ed22e1322884cdabc9780070dc/resolve.conf",
    "HostnamePath": "/home/jonpitsaf/.local/share/docker/containers/1192d8ecfbb3662d028542115dff5d44a80843ed22e1322884cdabc9780070dc/hostname",
    "HostsPath": "/home/jonpitsaf/.local/share/docker/containers/1192d8ecfbb3662d028542115dff5d44a80843ed22e1322884cdabc9780070dc/hosts",
    "LogPath": "/home/jonpitsaf/.local/share/docker/containers/1192d8ecfbb3662d028542115dff5d44a80843ed22e1322884cdabc9780070dc/1192d8ecfbb3662d028542115dff5d44a80843ed22e1322884cdabc9780070dc.json.log",
    "Name": "/Jonas1",
    "RestartCount": 0,
  }
]
```

Recuerda ajustar los comandos según tus necesidades específicas y tomar capturas de pantalla para documentar visualmente el proceso y los resultados de cada comando. Esto proporcionará a



los alumnos una práctica completa y variada en el manejo de contenedores Docker.

