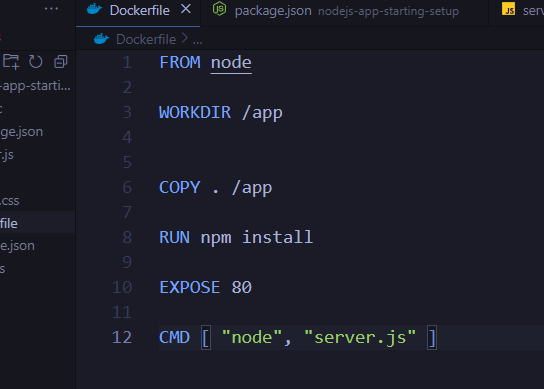
**Laboratorio 4A.**

**Crear un archivo Dockerfile. Operaciones sobre contenedores.**

**Paso 1.** Descargamos la carpeta comprimida con los archivos.

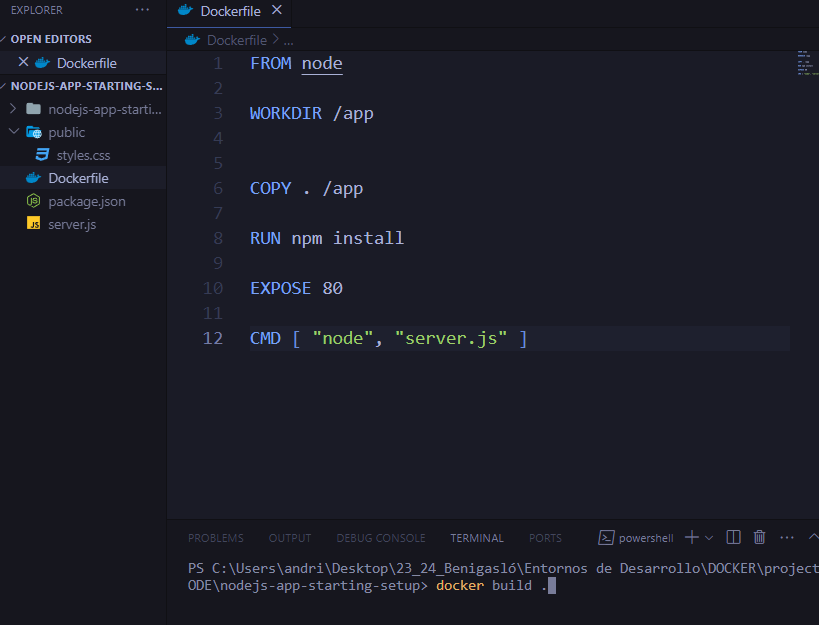
Creamos el **Dockerfile** sin la extensión.



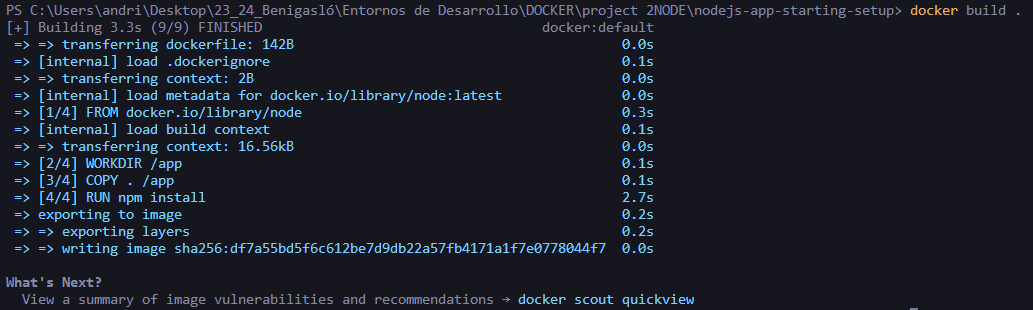
**Paso 2.** Creamos la imagen sobre el Dockerfile:

Utilizamos el comando ya conocido:

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

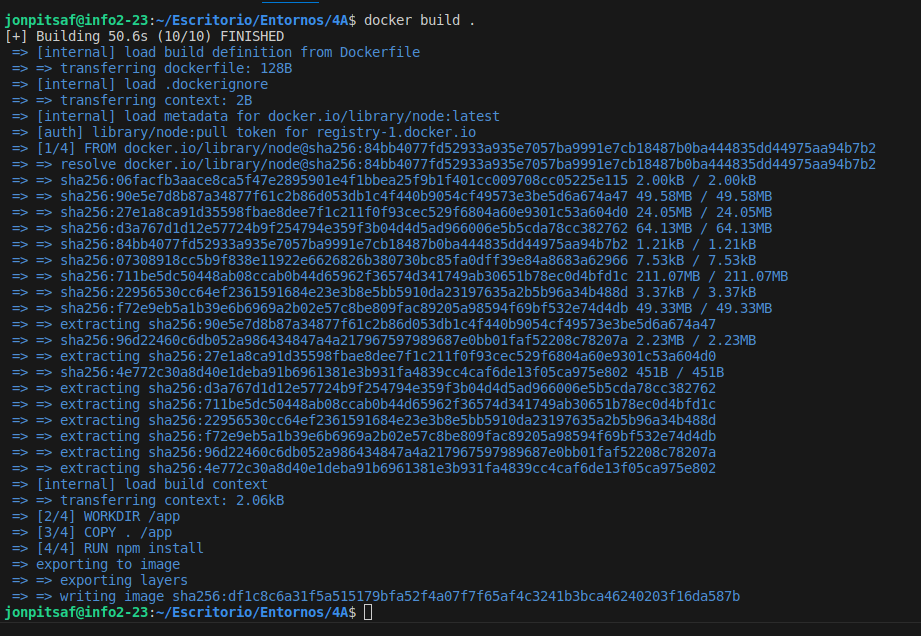


Resultado:



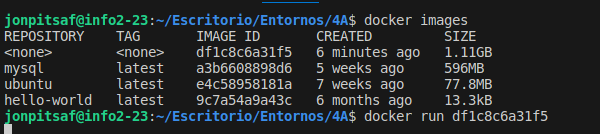
Podemos observar los pasos que hemos especificado en el Dockerfile, se han ejecutado sin errores.

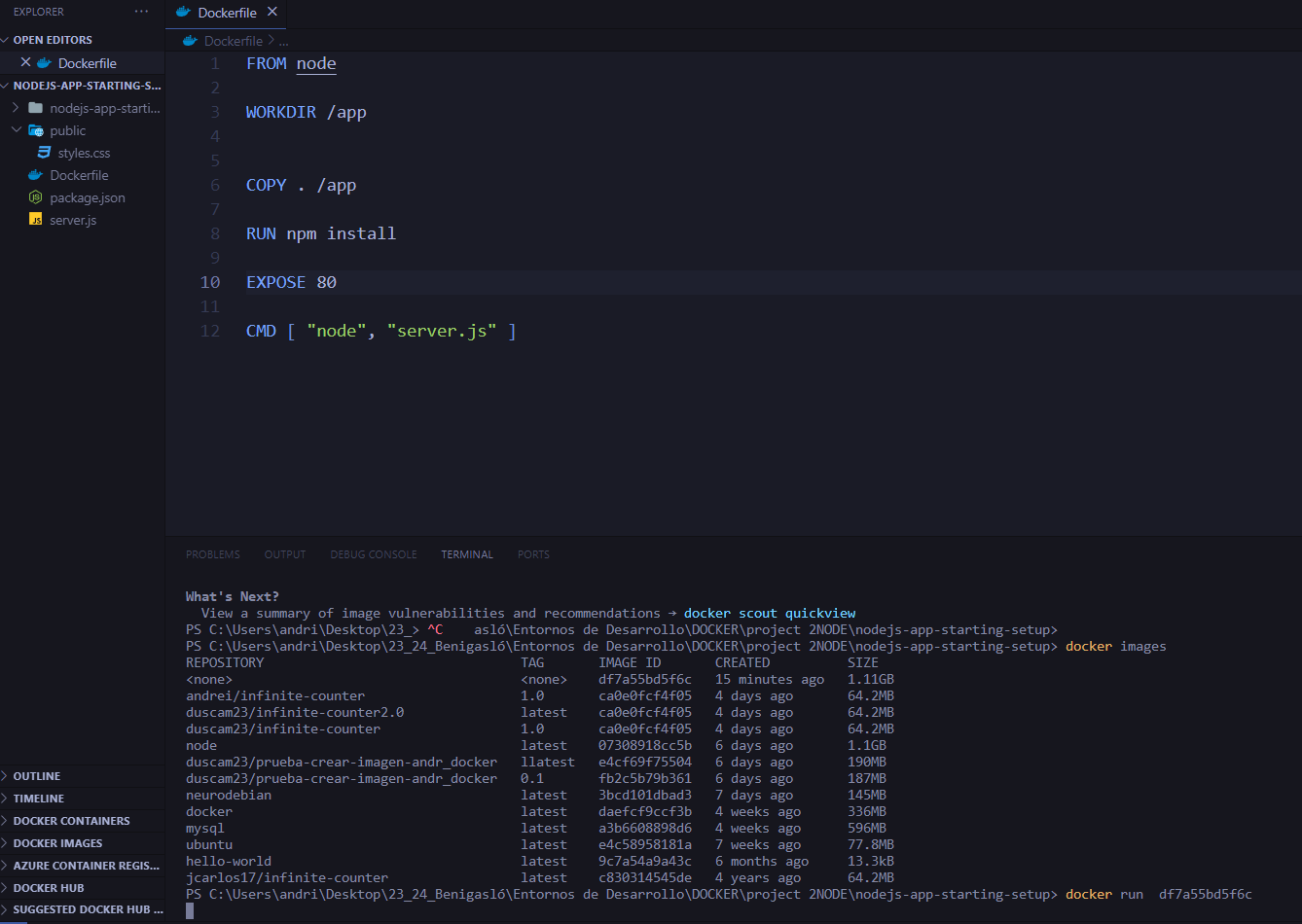
Paso 3. Buscamos el nombre de la imagen y creamos una instancia un proceso, es decir un contendor. Por lo tanto, copiamos el ID de la imagen y en la terminal de visual estudio code escribimos:

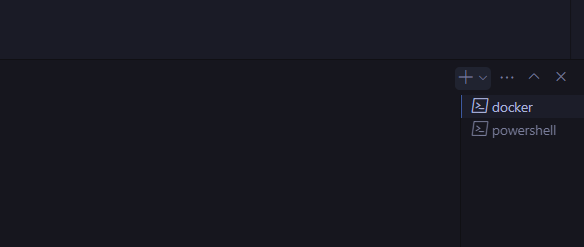


*docker run df7a*

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



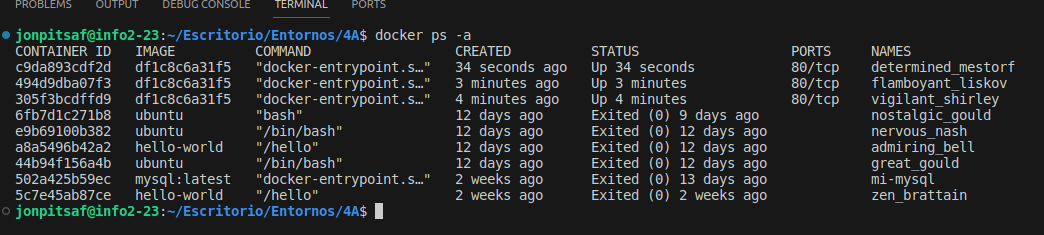


Paso 4. Abrimos un otro terminal en la parte derecha de la terminal VSC y configuramos los surgentes pasos .

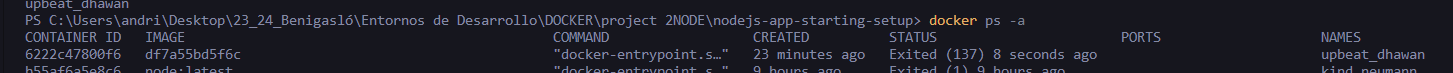


**Paso 5.** Para ver todos los procesos activos, en la terminal nueva escribimos el comando docker ps -a

Ahora el siguiente paso es parar nuestro contenedor. Lo haremos con el comando



*docker stop nombre del contenedor o ID*



**

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

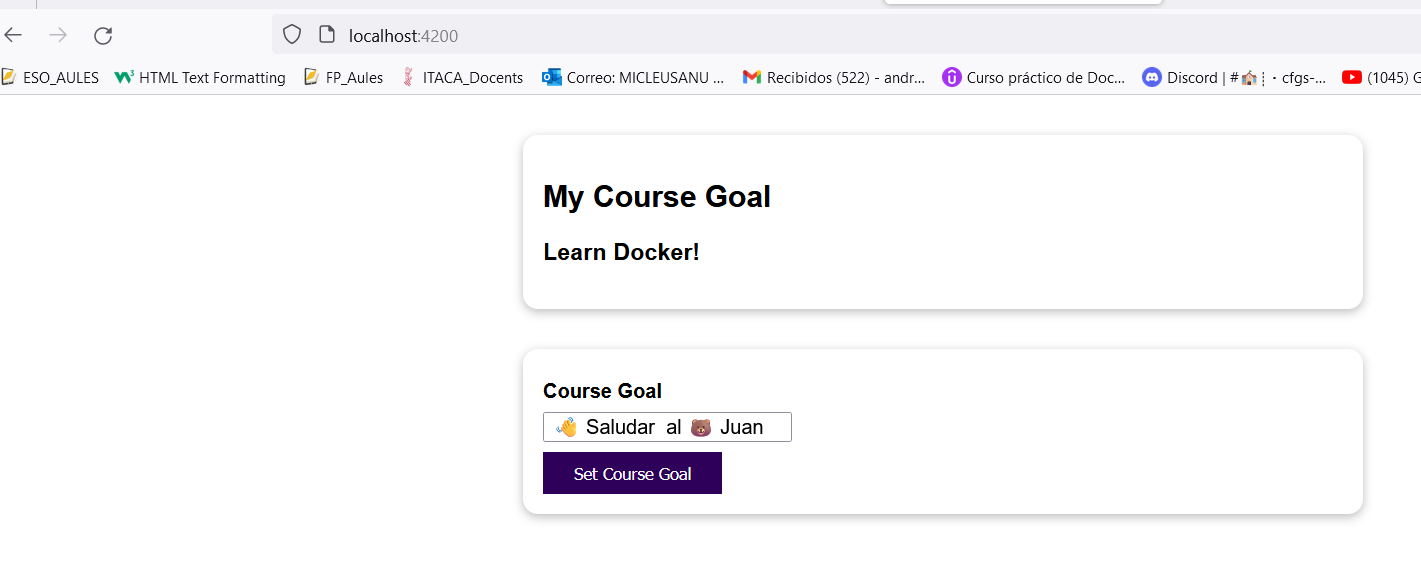
**Paso 6**. Si buscamos ver la respuesta en nuestro localhost 80, el resultado nos drá un error. Necesitamos realizar unos pasos opcionales. Volver a ejecutar el contenedor, pero necesitamos agregar una bandera adicional. Es decir:

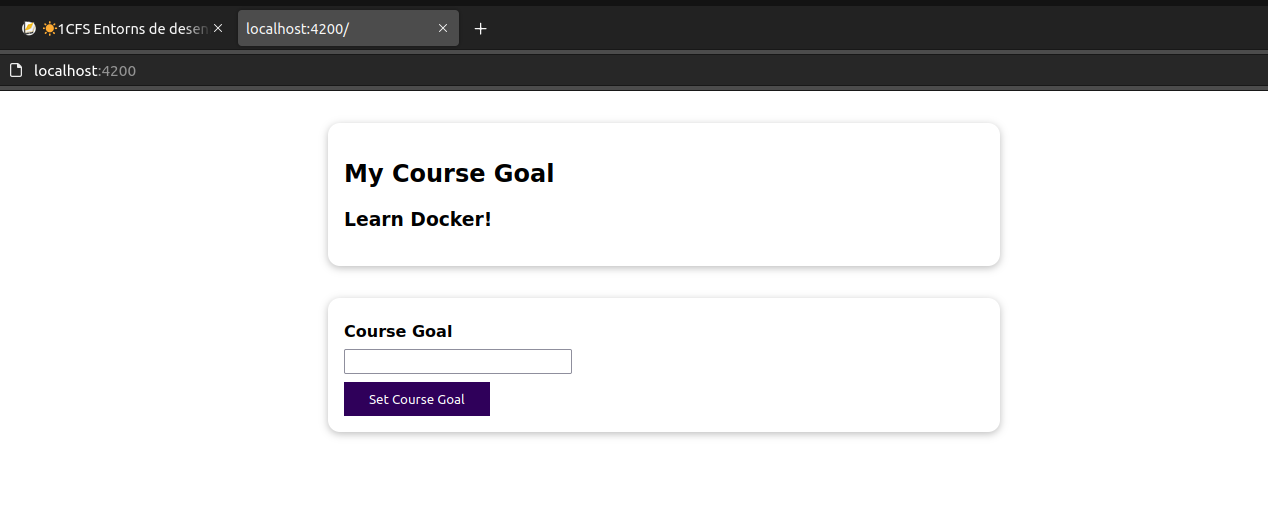
Añadimosla *bandera -p delante del nombre contenedor.*

*docker run -p 4200:80 df7a55bd5f6c*

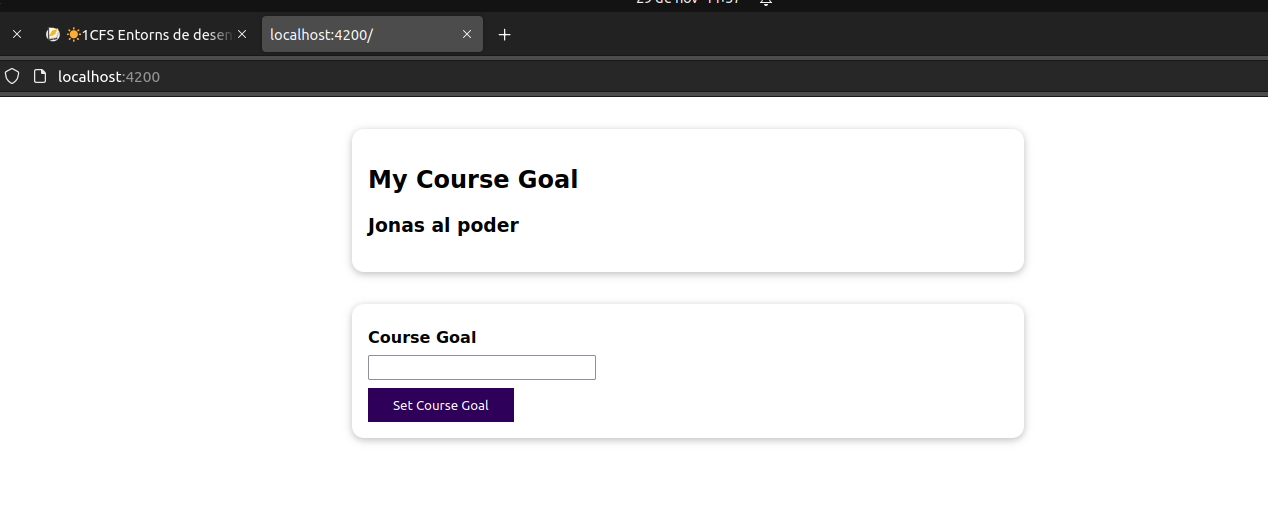
observamos en localhost 4200 respuesta: http://localhost:4200/







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.

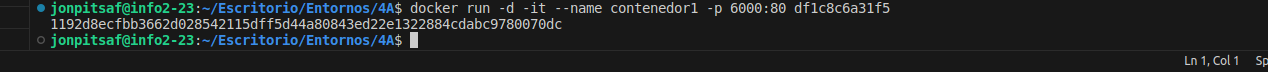


**Parte B. Operaciones y gestiones con los contenedores.**

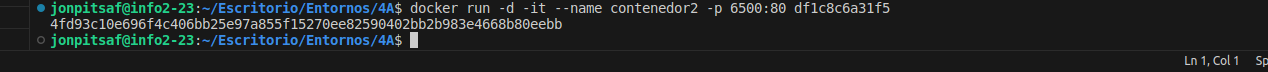
**Paso B1**. Crear Contenedores con Puertos Locales Específicos y Terminal Interactiva:

```**terminal**

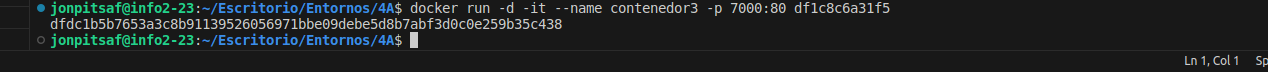
docker run -d -it --name contenedor1 -p 6000:puerto\_interno nombre\_de\_tu\_imagen



docker run -d -it --name contenedor2 -p 6500:puerto\_interno nombre\_de\_tu\_imagen



docker run -d -it --name contenedor3 -p 7000:puerto\_interno nombre\_de\_tu\_imagen



```

Asegúrate de reemplazar "puerto\_interno" ***busca en dockerhub*** [***https://hub.docker.com/layers/library/node/lts-slim/images/sha256-18aacc7993a16f1d766c21e3bff922e830bcdc7b549bbb789ceb7374a6138480?context=explore***](https://hub.docker.com/layers/library/node/lts-slim/images/sha256-18aacc7993a16f1d766c21e3bff922e830bcdc7b549bbb789ceb7374a6138480?context=explore)

***, o en***

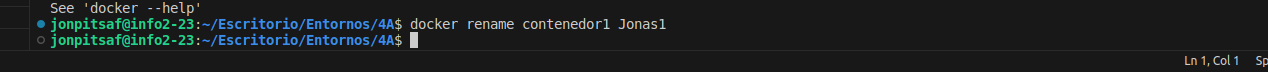
[***https://es.stackoverflow.com/search?q=puerto+expose+node***](https://es.stackoverflow.com/search?q=puerto+expose+node)

***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



```

**Paso B**3. \*\*Adjuntar un Archivo al Contenedor en Ejecución:\*\*

``` **terminal**

docker cp archivo\_adjunto.txt contenedor2:/ruta/destino

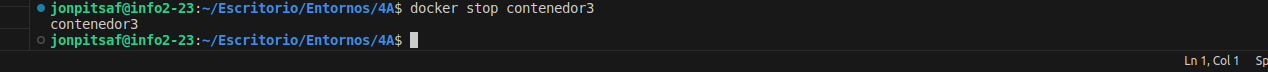
```

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

``` **terminal**

docker stop contenedor3

```

****

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

``` **terminal**

docker exec -it contenedor1 /bin/bash



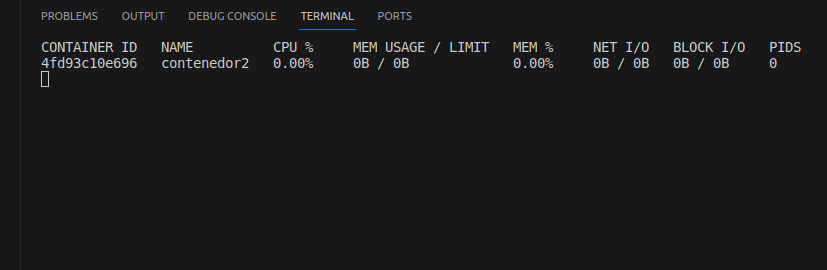
```

Esto abrirá una terminal interactiva en el contenedor1.

**Paso B6**. Sugerencia Adicional: Verificar Estadísticas del Contenedor:

``` **terminal**

docker stats contenedor2



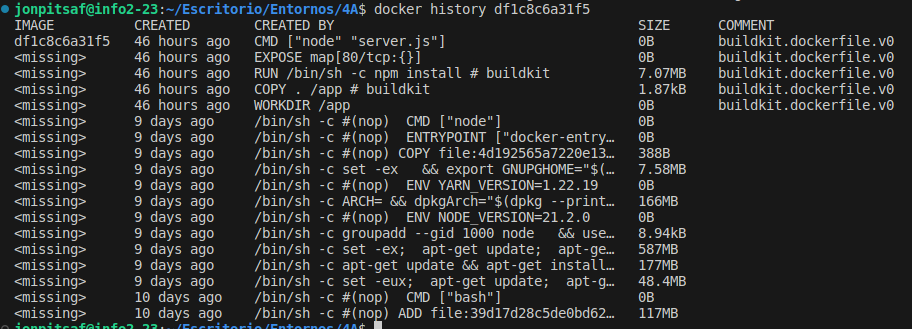
```

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



```

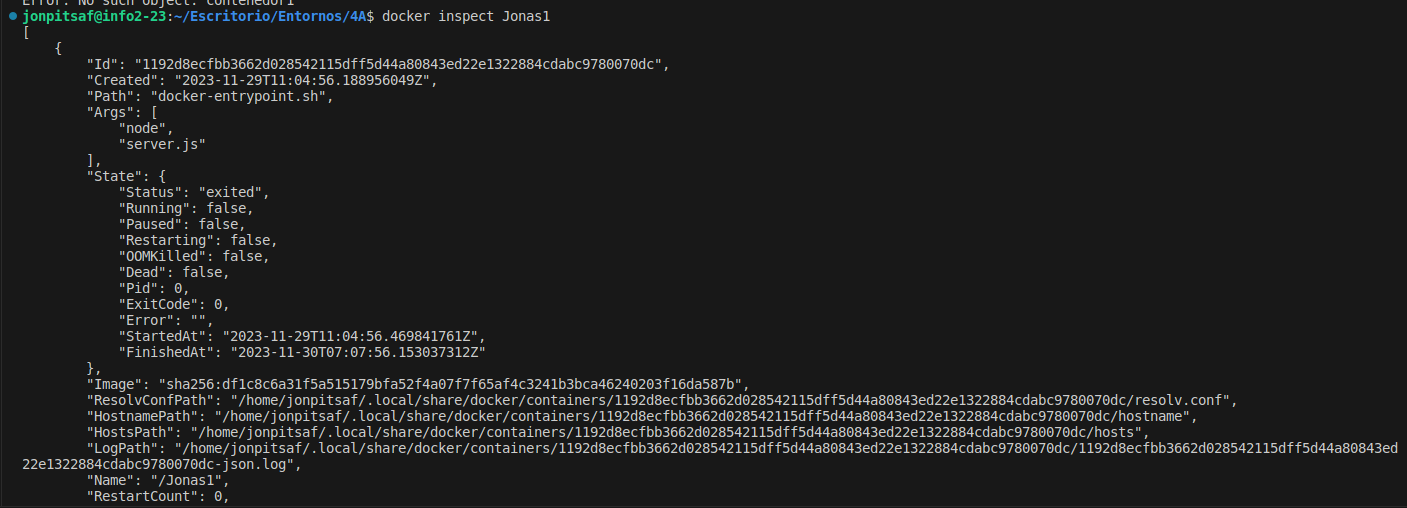
Este comando mostrará el historial de capas de la imagen Docker.

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

``` **terminal**

docker inspect contenedor1

```



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.