

Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e
Ingenierías

Computación Tolerante a Fallas

Mtro: Michell Lopez Franco

Juan Antonio Pérez Juárez

INCO

215660996

Docker

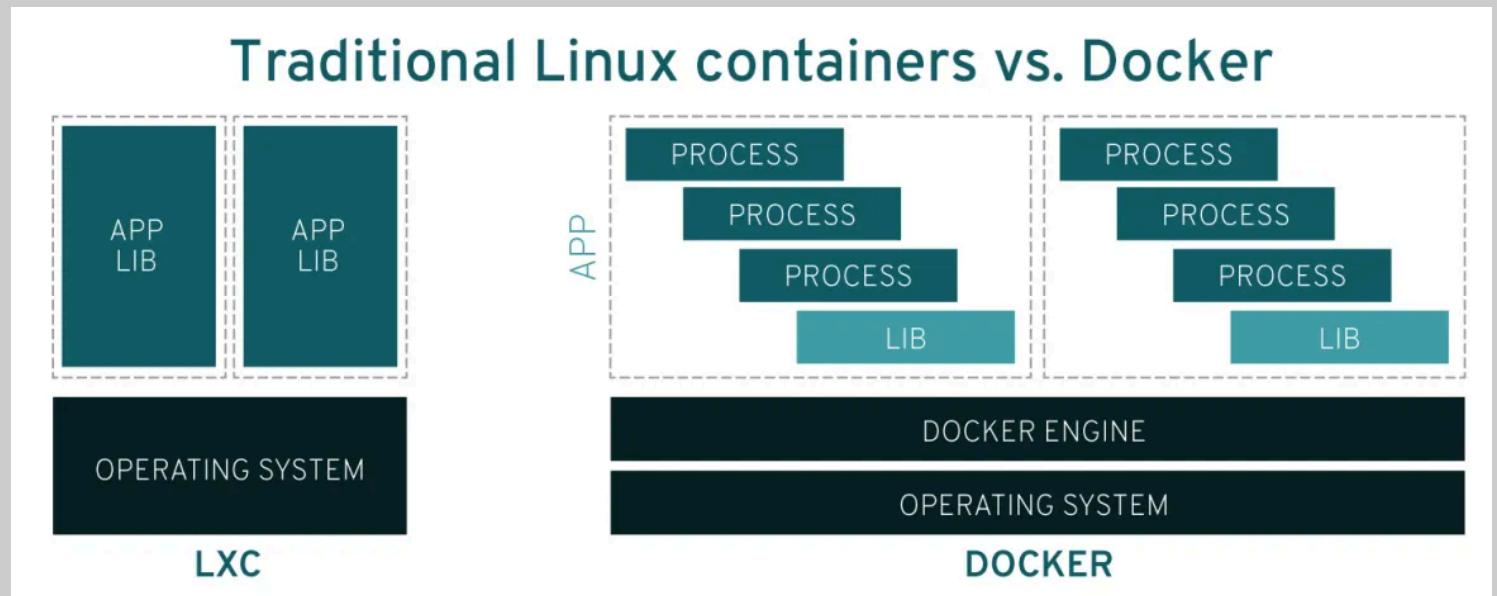
Introducción:

La adopción de Docker ha crecido exponencialmente en los últimos años, convirtiéndose en un estándar de facto en la industria tecnológica. Desde startups hasta grandes empresas, organizaciones de todos los tamaños han adoptado esta tecnología para optimizar sus procesos de desarrollo y mejorar la eficiencia operativa. La containerización no es solo una tendencia pasajera, sino una revolución fundamental en la forma en que construimos y desplegamos software.

Desarrollo:

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

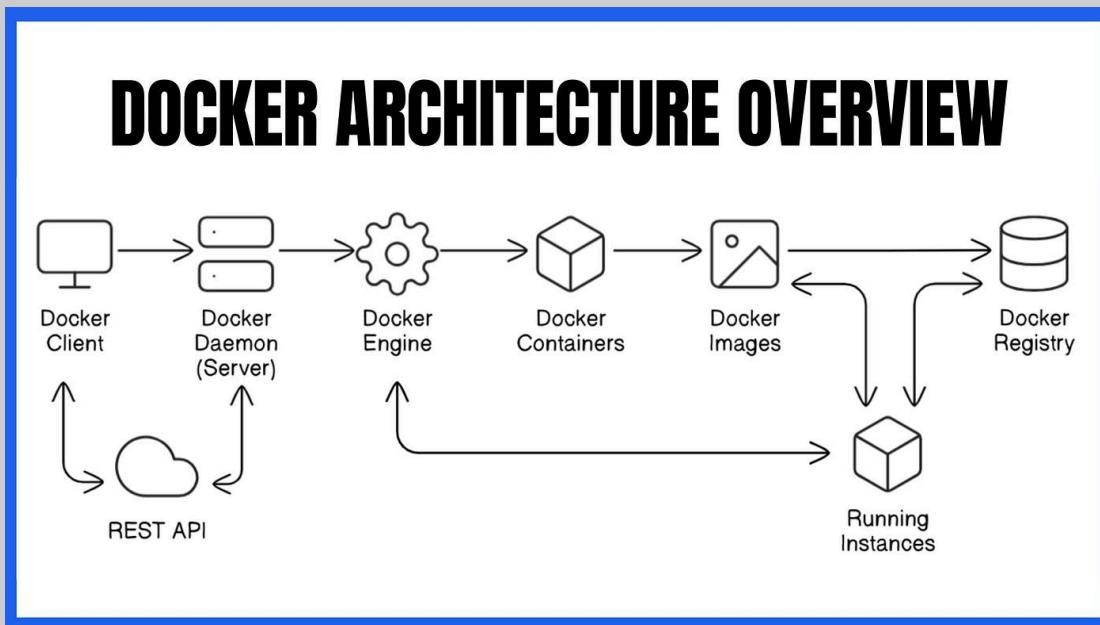
Docker utiliza características de aislamiento de recursos del kernel Linux, tales como cgroups y espacios de nombres (namespaces) para permitir que "contenedores" independientes se ejecuten dentro de una sola instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales.



El soporte del kernel Linux para los espacios de nombres aísla la vista que tiene una aplicación de su entorno operativo, incluyendo árboles de proceso, red, ID de usuario y sistemas de archivos montados, mientras que los cgroups del kernel proporcionan aislamiento de recursos, incluyendo la CPU, la memoria, el bloqueo de E/S y de la red. Desde la versión 0.9, Docker incluye la biblioteca libcontainer como su propia manera de utilizar directamente las facilidades de virtualización que ofrece el kernel Linux, además de utilizar las interfaces abstractas de virtualización mediante libvirt, LXC (Linux Containers) y systemd-nspawn.

De acuerdo con la firma analista de la industria 451 Research, "Docker es una herramienta que puede empaquetar una aplicación y sus dependencias en un contenedor virtual que se puede ejecutar en cualquier servidor Linux. Esto ayuda a permitir la flexibilidad y portabilidad en donde la aplicación se puede ejecutar, ya sea en las instalaciones físicas, la nube pública, nube privada, etc.

Usar Docker para crear y gestionar contenedores puede simplificar la creación de sistemas altamente distribuidos, permitiendo que múltiples aplicaciones, las tareas de los trabajadores y otros procesos funcionen de forma autónoma en una única máquina física o en varias máquinas virtuales. Esto permite que el despliegue de nodos se realice a medida que se dispone de recursos o cuando se necesiten más nodos, lo que permite una plataforma como servicio (PaaS - Platform as a Service) de estilo de despliegue y ampliación de los sistemas como Apache Cassandra, MongoDB o Riak. Docker también simplifica la creación y el funcionamiento de las tareas de carga de trabajo o las colas y otros sistemas distribuidos.



Solomon Hykes comenzó Docker como un proyecto interno dentro de dotCloud, empresa enfocada a una plataforma como un servicio (PaaS), con las contribuciones iniciales de otros ingenieros de dotCloud, incluyendo a Andrea Luzzardi y Francois-Xavier Bourlet. Jeff Lindsay también participó como colaborador independiente. Docker representa una evolución de la tecnología patentada de dotCloud, que es a su vez construida sobre proyectos de código abierto anteriores como Cloudlets.

Docker fue liberado como código abierto en marzo de 2013. El 13 de marzo de 2014, con el lanzamiento de la versión 0.9, Docker dejó de utilizar LXC como el entorno de ejecución por defecto y lo reemplazó con su propia biblioteca, libcontainer, escrita en Go. El 13 de abril de 2015, el proyecto tenía más de 20 700 estrellas de GitHub (haciéndolo uno de los proyectos con más estrellas de GitHub, en 20.^a posición), más de 4 700 bifurcaciones (forks), y casi 900 colaboradores.

Un análisis en 2018 mostró las siguientes organizaciones como las principales contribuyentes de Docker: Red Hat (mayores contribuyentes, aún más que el equipo de Docker en sí), el equipo de Docker, Microsoft, IBM, Google, Cisco Systems y Amadeus IT Group.



Ventajas de los contenedores Docker

Modularidad

El enfoque de Docker sobre la organización en contenedores se centra en la capacidad de separar una parte de la aplicación para actualizarla o repararla, sin necesidad de deshabilitarla por completo. Además de aprovechar este modelo basado en los microservicios, puede intercambiar procesos entre varias aplicaciones casi de la misma forma en que funciona la arquitectura orientada a los servicios (SOA).

Capas y control de versiones de imágenes

Cada archivo de imagen Docker está compuesto por varias capas que conforman una sola imagen. Cuando un usuario especifica un comando, como ejecutar o copiar, la imagen cambia, y se crea una capa nueva.

Docker reutiliza las capas para agilizar el diseño de los contenedores nuevos. Los cambios intermedios se comparten entre las imágenes para mejorar aún más la agilidad, el tamaño y la eficiencia. El control de versiones también es propio de la creación de capas: el registro incorporado de los cambios le brinda el control total de las imágenes de contenedores cada vez que se produce una modificación.

Restauración

Uno de los mayores beneficios de las capas es la capacidad de restauración. Todas las imágenes cuentan con capas. Si no le gusta la iteración actual de una imagen, puede restaurarla a una versión anterior. Esto respalda el enfoque de desarrollo ágil y permite lograr la integración e implementación continuas (CI/CD) desde la perspectiva de las herramientas.

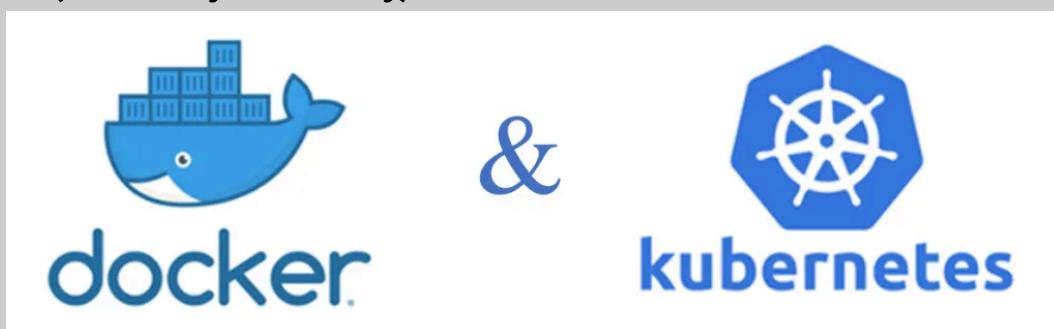
Implementación rápida

Antes, se necesitaban varios días para poner en marcha un sistema de hardware nuevo, implementarlo y ponerlo a disposición de los usuarios, lo cual implicaba un esfuerzo y un costo abrumador. Con los contenedores basados en Docker, la implementación se puede realizar en cuestión de segundos. Cada proceso se encuentra en un contenedor distinto, por lo que puede compartirlos con aplicaciones nuevas rápidamente. Además, ya que no es necesario iniciar el sistema operativo para agregar o trasladar un contenedor, los tiempos de implementación son mucho más cortos, y puede crear datos y eliminar aquellos que generen los contenedores de manera fácil y rentable, sin preocupaciones.

En definitiva, la tecnología Docker tiene un enfoque más detallado y controlable, que se basa en los microservicios y prioriza la eficiencia.

Límites para el uso de Docker

Docker, por sí solo, puede gestionar contenedores individuales. Si comienza a utilizar cada vez más contenedores y aplicaciones que se alojan en ellos con cientos de elementos, se dificultará su gestión y organización. En algún momento, deberá cambiar el enfoque y agrupar los contenedores para prestar los servicios, como las redes, la seguridad y la telemetría, entre otros, en todos los contenedores. Allí entra en juego **Kubernetes. (Directed by Michael Bay)**



Para el desarrollo de esta actividad primero implementamos una imagen de ejemplo siguiendo los comandos que nos publicó el profe en classroom.

Primero que nada debemos instalar Docker Desktop, lo cual es muy sencillo desde el portal de Docker.

<https://www.docker.com/products/docker-desktop/>

Y es importante que seleccionemos la opción de agregar al PATH para funciones de variables de entorno.

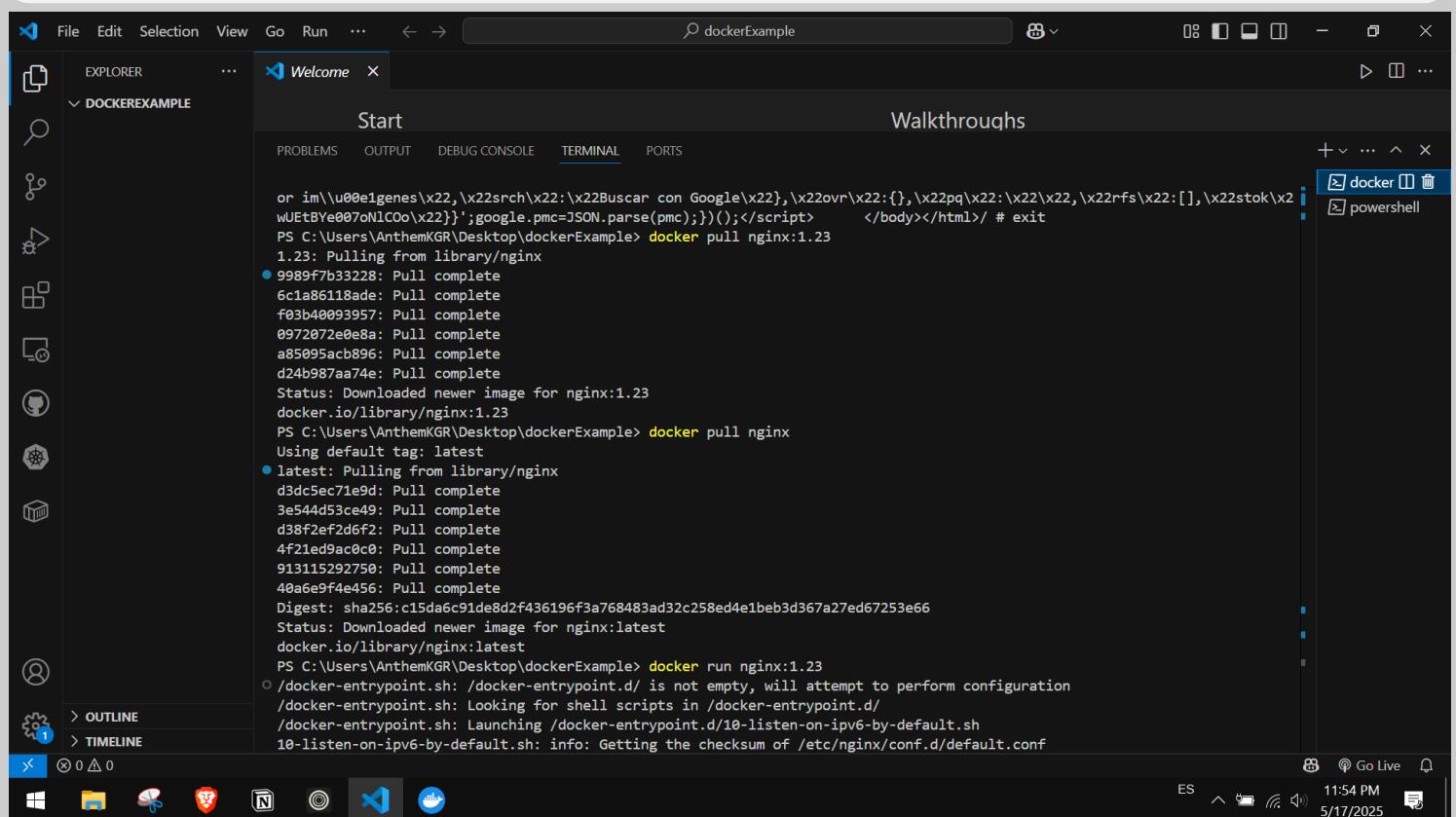
Cuando lo tengamos descargado lo ejecutamos para que corra en segundo plano.

En mi caso cree una carpeta en mi escritorio en el cual almacenaré el Docker y los archivos para este reporte.

En una terminal, que yo suelo utilizar la terminal de VS code, ejecutamos los siguientes comandos.

```
Unset
```

```
docker images
docker ps
docker pull alpine:3.18.4
docker run -it alpine:3.18.4 sh
apk update
apk upgrade
apk add curl
curl www.google.com
exit
docker pull nginx:1.23
docker pull nginx
docker run nginx:1.23
```



The screenshot shows the VS Code interface with the terminal tab active. The terminal window displays the following command history:

```
PS C:\Users\AnthemKGR\Desktop\dockerExample> docker pull nginx:1.23
1.23: Pulling from library/nginx
9989f7b33228: Pull complete
6c1a86118ade: Pull complete
f03b40093957: Pull complete
0972072e0e8a: Pull complete
a85095acb896: Pull complete
d24b987a7a74: Pull complete
Status: Downloaded newer image for nginx:1.23
docker.io/library/nginx:1.23
PS C:\Users\AnthemKGR\Desktop\dockerExample> docker pull nginx
Using default tag: latest
● latest: Pulling from library/nginx
d3dc5ec71e9d: Pull complete
3e544d53ce49: Pull complete
d38f2ef2df6f: Pull complete
4f21ed9ac0c0: Pull complete
913115292750: Pull complete
40a6e9f4e456: Pull complete
Digest: sha256:c15da6c91de8d2f436196f3a768483ad32c258ed4e1beb3d367a27ed67253e66
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
PS C:\Users\AnthemKGR\Desktop\dockerExample> docker run nginx:1.23
○ /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
```

En otra terminal ejecutaremos lo siguiente:

Unset

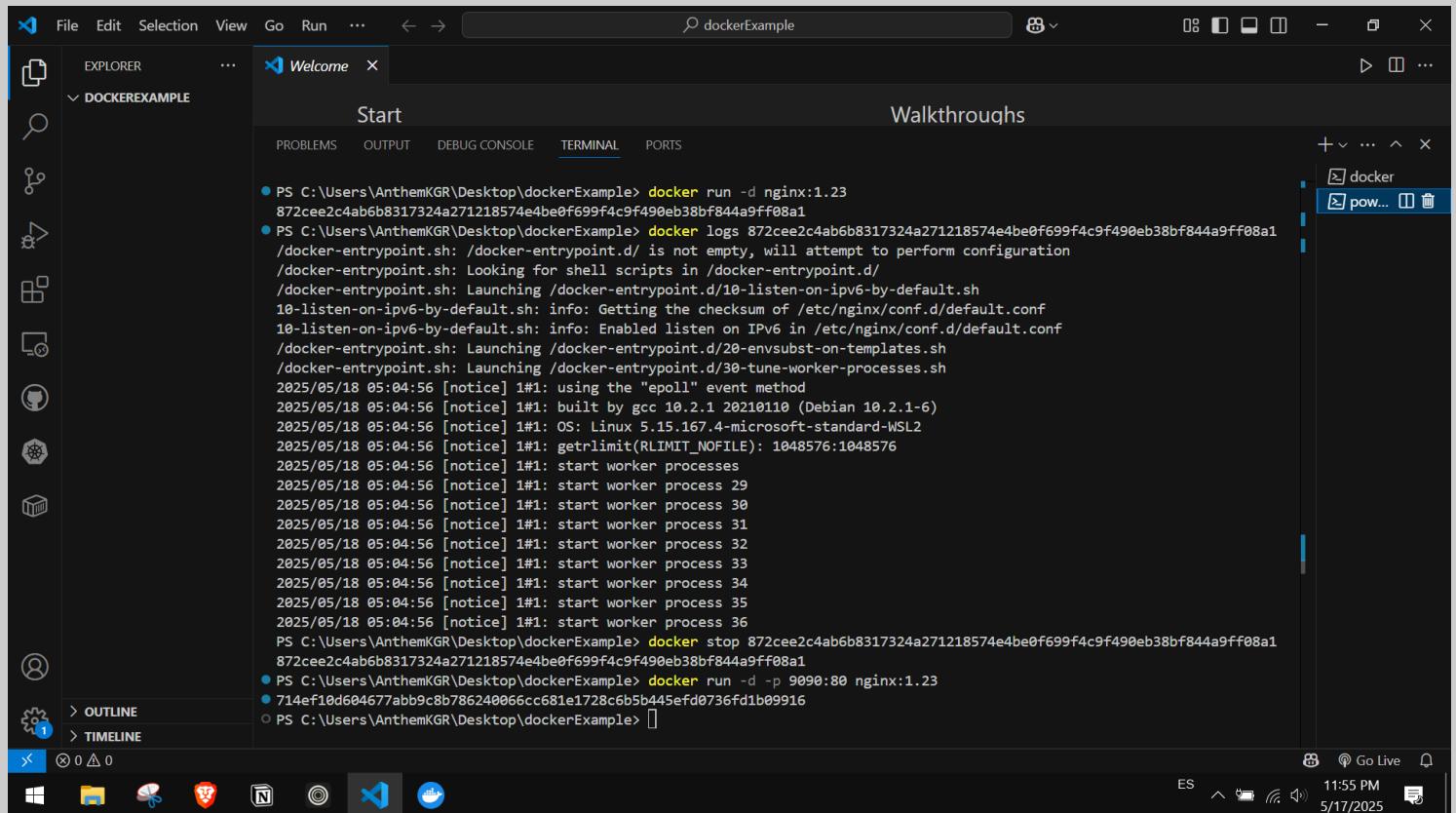
```
docker ps  
docker run -d nginx:1.23  
docker logs {id}
```

Ejemplo:

```
Unset  
docker logs 7250a9f1d7b9  
  
docker stop 7250a9f1d7b9
```

Exponemos un puerto:

```
Unset  
docker run -d -p 9090:80 nginx:1.23
```

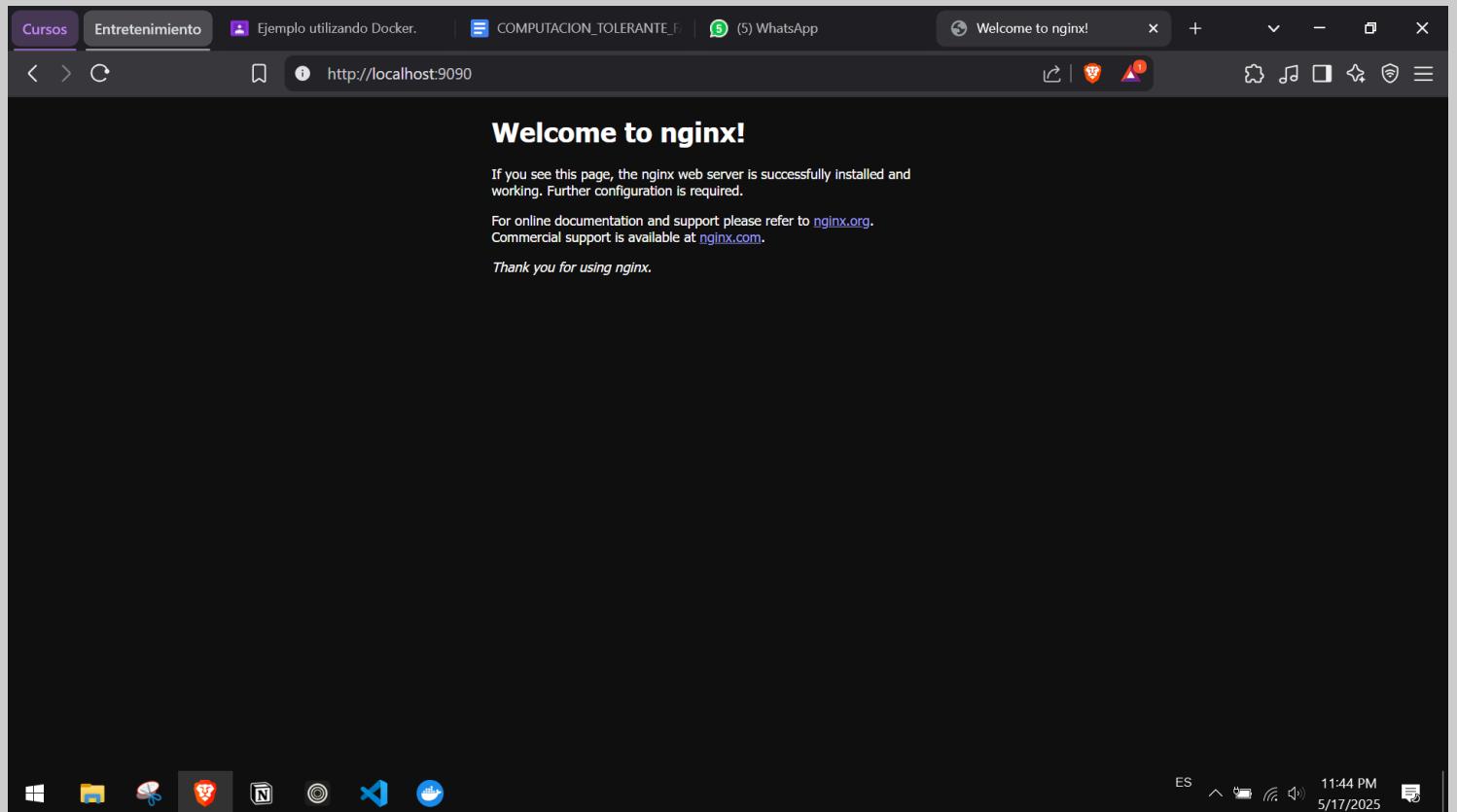


The screenshot shows the Microsoft Visual Studio Code (VS Code) interface. The title bar says "dockerExample". The left sidebar has icons for Explorer, Search, Files, and others, with "DOCKEREXAMPLE" selected. The main area has tabs for "Start" and "Walkthroughs". Below them are "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL", and "PORTS". The "TERMINAL" tab is active, displaying the following Docker logs:

```
PS C:\Users\AnthemKGR\Desktop\dockerExample> docker run -d nginx:1.23  
872cee2c4ab6b8317324a271218574e4be0f699f4c9f490eb38bf844a9ff08a1  
● PS C:\Users\AnthemKGR\Desktop\dockerExample> docker logs 872cee2c4ab6b8317324a271218574e4be0f699f4c9f490eb38bf844a9ff08a1  
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration  
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/  
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh  
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf  
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf  
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh  
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh  
2025/05/18 05:04:56 [notice] 1#1: using the "epoll" event method  
2025/05/18 05:04:56 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)  
2025/05/18 05:04:56 [notice] 1#1: OS: Linux 5.15.167.4-microsoft-standard-WSL2  
2025/05/18 05:04:56 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576  
2025/05/18 05:04:56 [notice] 1#1: start worker processes  
2025/05/18 05:04:56 [notice] 1#1: start worker process 29  
2025/05/18 05:04:56 [notice] 1#1: start worker process 30  
2025/05/18 05:04:56 [notice] 1#1: start worker process 31  
2025/05/18 05:04:56 [notice] 1#1: start worker process 32  
2025/05/18 05:04:56 [notice] 1#1: start worker process 33  
2025/05/18 05:04:56 [notice] 1#1: start worker process 34  
2025/05/18 05:04:56 [notice] 1#1: start worker process 35  
2025/05/18 05:04:56 [notice] 1#1: start worker process 36  
PS C:\Users\AnthemKGR\Desktop\dockerExample> docker stop 872cee2c4ab6b8317324a271218574e4be0f699f4c9f490eb38bf844a9ff08a1  
872cee2c4ab6b8317324a271218574e4be0f699f4c9f490eb38bf844a9ff08a1  
● PS C:\Users\AnthemKGR\Desktop\dockerExample> docker run -d -p 9090:80 nginx:1.23  
714ef10d604677ab9c8b786240066cc681e1728c6b5b445efd0736fd1b09916  
○ PS C:\Users\AnthemKGR\Desktop\dockerExample>
```

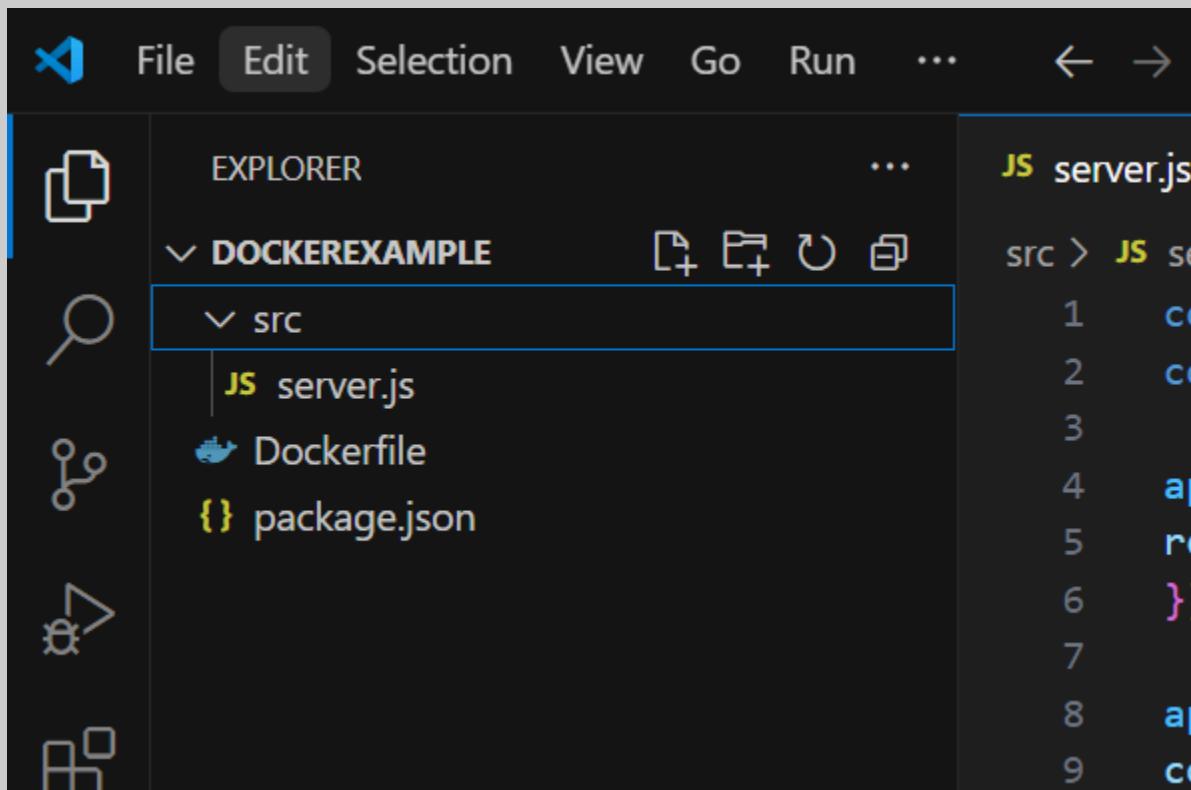
Abre tu navegador y pon localhost:9090 mostrara un mensaje de bienvenida el nginx

La primera parte de los comandos es echar a andar un docker con Nginx, exponiendo un puerto y entrando desde el navegador, que se muestre la siguiente pantalla es como un checkpoint.



Así es como se debe ver cuando llegamos a ese punto.

Siguiendo las indicaciones debemos generar varios archivos y nuestro directorio debe verse algo como esto:



```
1  const express = require('express');
2  const app = express();
3  const port = 3001;
4  app.get('/', (req, res) => {
5    res.send('Hello world');
6  });
7  app.listen(port, () => {
8    console.log(`App listening on port ${port}`);
9  });
});
```

En el cual el archivo server.js contendrá lo siguiente:

```
JavaScript
const express = require('express');
const app = express();

app.get('/', (req, res) =>{
res.send("Welcome to my awesome app!");
});

app.listen(3000, function () {
console.log("app listening on port 3000");
});
```

El archivo package.json contendrá:

```
JavaScript
{
"name": "my-app",
"version": "1.0",
"dependencies": {
"express": "4.18.2"
}
}
```

Y para finaliza el archivo Dockerfile contendrá:

```
Unset
FROM node:19-alpine

COPY package.json /app/
COPY src /app/

WORKDIR /app

RUN npm install

CMD [ "node", "server.js" ]
```

Construimos con la siguiente instrucción:

```
Unset
docker build -t node-app:1.0 .
```

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The title bar reads "File Edit Selection View Go Run ... dockerExample". The left sidebar has icons for Explorer, Search, Docker, and Terminal. The Explorer view shows a project structure with a "Dockerfile" selected. The main area is a terminal window displaying the following output:

```
+ FullyQualifiedErrorId : CommandNotFoundException  
PS C:\Users\AnthemKGR\Desktop\dockerExample> docker build -t node-app:1.0 .  
● [+] Building 16.0s (11/11) FINISHED  
=> [internal] load build definition from Dockerfile  
=> => transferring dockerfile: 164B  
=> [internal] load metadata for docker.io/library/node:19-alpine  
=> [auth] library/node:pull token for registry-1.docker.io  
=> [internal] load .dockerignore  
=> => transferring context: 2B  
=> [1/5] FROM docker.io/library/node:19-alpine@sha256:8ec543d4795e2e85af924a24f8acb039792ae9fe8a42ad5b4bf4c27  
=> => resolve docker.io/library/node:19-alpine@sha256:8ec543d4795e2e85af924a24f8acb039792ae9fe8a42ad5b4bf4c27  
=> => sha256:47be83a79857fb67c4d144471b8301aeefba874971bfbaa0d12dc97ea135cfffe 4498 / 4498  
=> => sha256:f352bc07f19b43a8678cc8e8fe162ccb6193ead7af6dd366639a01402d1819e 2.34MB / 2.34MB  
=> => sha256:1197750296b3abe1d21ffbb3d3eae76df5ba887cf82c8e3284d267ccb2aa1724a 48.15MB / 48.15MB  
=> => extracting sha256:8a49fd2b36a5ff2bd8ec6a86c05b2922a0f7454579ecc07637e94df1d0639b6  
=> => extracting sha256:1197750296b3abe1d21ffbb3d3eae76df5ba887cf82c8e3284d267ccb2aa1724a  
=> => extracting sha256:f352bc07f19b43a8678cc8e8fe162ccb6193ead7af6dd366639a01402d1819e  
=> => extracting sha256:47be83a79857fb67c4d144471b8301ae6fb874971bfbaa60d12dc97ea135cfffe  
=> [internal] load build context  
=> => transferring context: 417B  
=> [2/5] COPY package.json /app/  
=> [3/5] COPY src /app/  
=> [5/5] RUN npm install  
=> exporting to image  
=> => exporting layers  
=> => exporting manifest sha256:94736c6124d6b5c5c52ce9f9b368feb89d06ee28e4ca5d5dbe11f9f68f20bd2b  
=> => exporting config sha256:9fc5c4c8ca9e743e3f08ab4238570a3158a689515fc41e80e709e4b2bbdbab1161  
=> => exporting attestation manifest sha256:0d0723c5b0dcafdf258bedeb13d40b11d40016b7063760ff7e19a85686f66c  
=> => exporting manifest list sha256:0f5f81a324a583076bd917205d10b2d6384aec16183213b8bceb9d7e84a4f6  
=> => naming to docker.io/library/node-app:1.0  
PS C:\Users\AnthemKGR\Desktop\dockerExample> -t  
-t : The term '-t' is not recognized as the name of a cmdlet, function, script file, or operable program. Check
```

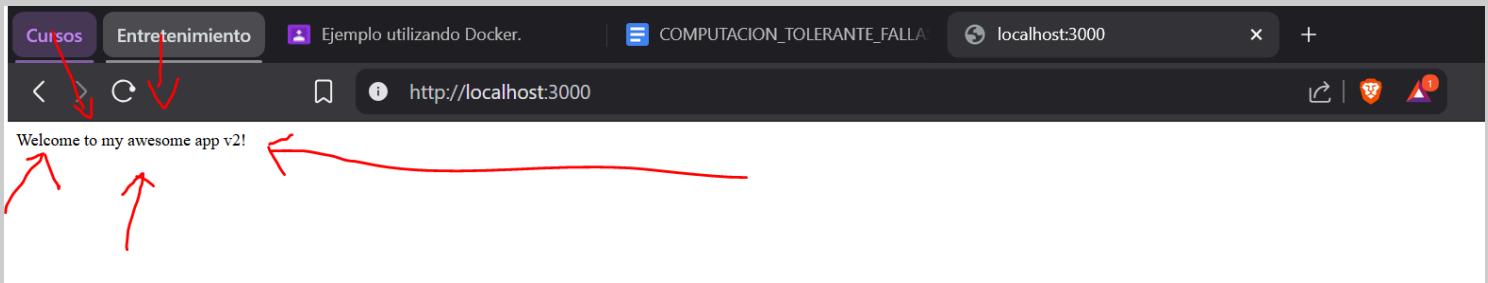
Por lo que si visitamos: <http://localhost:3000/>, tendremos que ver esto:



Detuvimos la ejecución de nuestra app maravillosa e insertamos la siguiente modificación en el archivo **server.js**:

```
JavaScript
app.get('/', (req, res) =>{
  res.send("Welcome to my awesome app v2!");
});
```

Por lo que si guardamos y volvemos a reiniciar la app deberemos ver un cambio en nuestro <http://localhost:3000>



Conclusión:

Joder que buena actividad, joder, esto sí es cine.

A pesar de que no generé ninguna app, como ya tenía los conocimientos de la actividad pasada se me hizo super sencillo el poder desplegar la app, y me gustó.

Pude hacer todo yo sin ayuda de la IA, y eso es un logro muy interesante para mí, a pesar que no dependo de la IA para programas sinó que la uso para detalles estéticos en su mayoría, hacer esto sin abrir ninguna me ayudó con mi confianza como programador.

Referencias:

colaboradores de Wikipedia. (2025, April 27). Docker (software). Wikipedia, La Enciclopedia Libre. [https://es.wikipedia.org/wiki/Docker_\(software\)](https://es.wikipedia.org/wiki/Docker_(software))

¿Qué es Docker y cómo funciona? Ventajas de los contenedores Docker. (n.d.). <https://www.redhat.com/es/topics/containers/what-is-docker>

Dock, M. (2025, May 16). Docker Desktop: The #1 containerization tool for developers | Docker. Docker. <https://www.docker.com/products/docker-desktop/>