

División de Tecnologías para la integración Ciber-humana
Departamento de Ciencias Computacionales



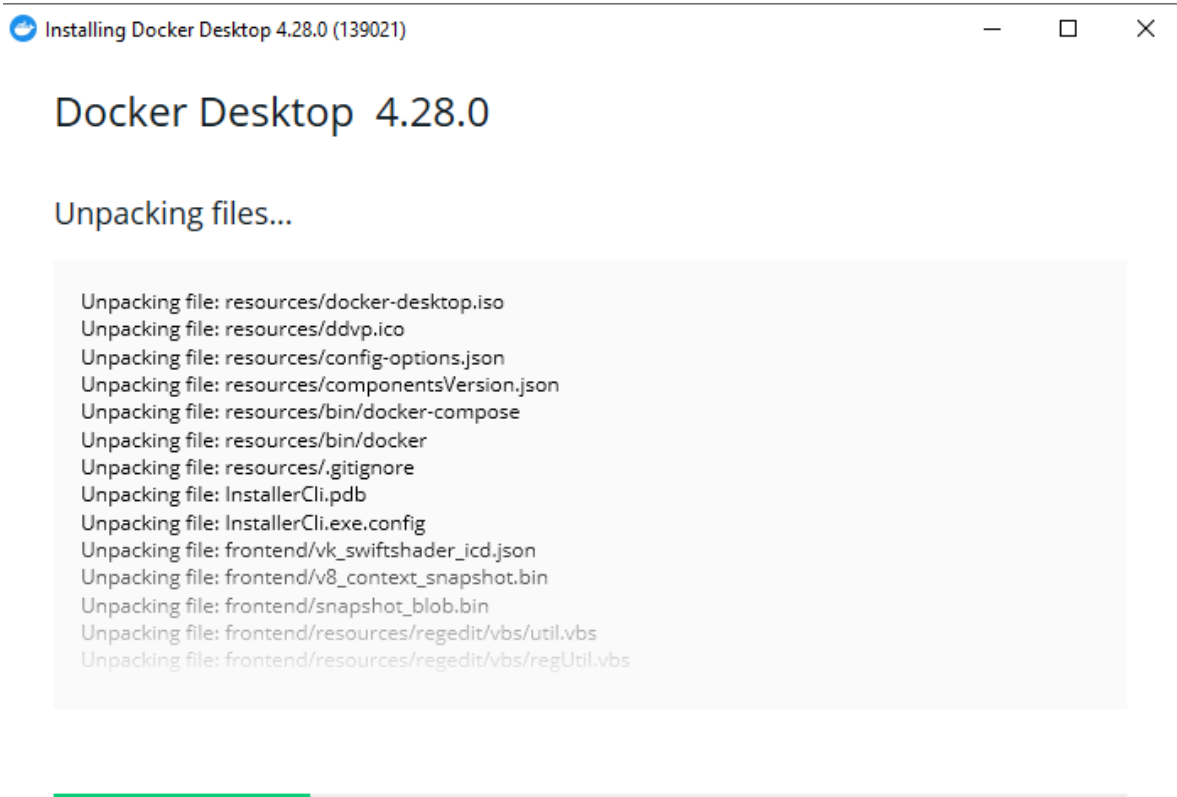
Tema: Ejemplo utilizando Docker

Mercado Hernandez Angel Gabriel

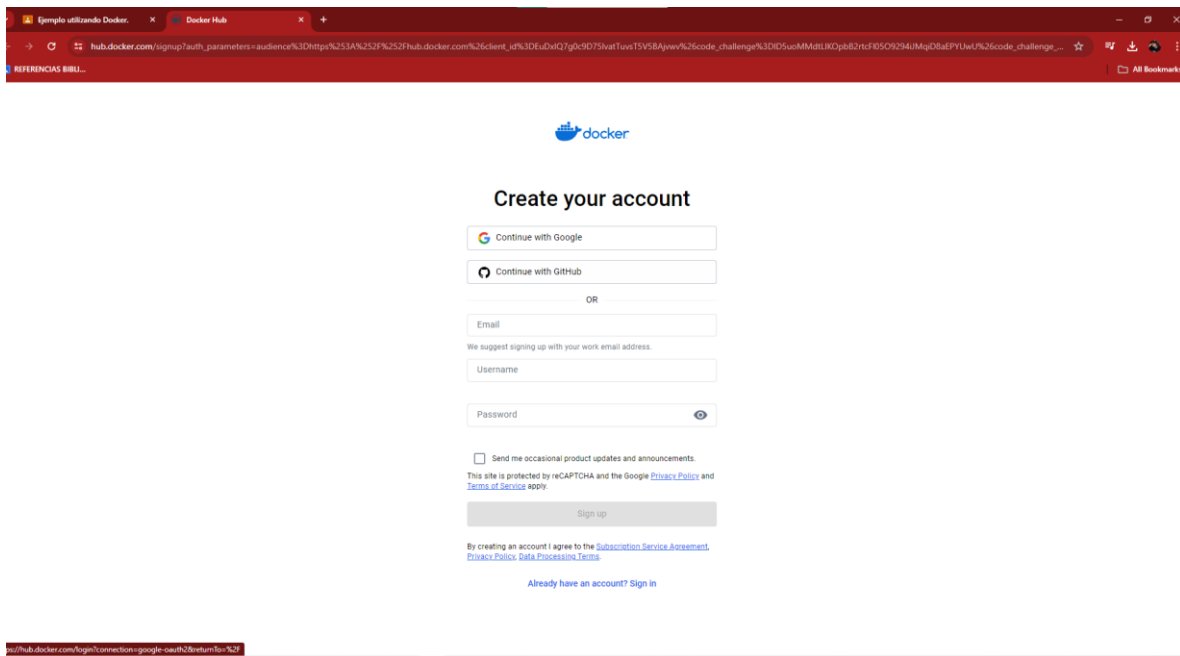
Computación tolerante a fallas. Sección D06. Lunes y Miércoles 11:00
AM – 12:55 AM

Lopez Franco Michel Emanuel

Para este primer paso descargue e instale Docker para la computadora



Aquí vincule mi cuenta de GitHub para más rapidez y mas eficacia



Ya despues sigue abrir la terminal para ejecutar los comandos

docker images

docker ps

e instalar Alpine

docker pull alpine:3.18.4

Dentro del contenedor Alpine, ejecuta los siguientes comandos para actualizar e instalar curl y luego probar la conexión a Google:

apk update

apk upgrade

apk add curl

curl www.google.com

exit

```

C:\Users\USER>docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
C:\Users\USER>docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS        NAMES
C:\Users\USER>docker pull alpine:3.18.4
3.18.4: Pulling from library/alpine
96526aa774ef: Pull complete
Digest: sha256:eace025e432126ce23f223450a0326fbebde39cdf496a85d8c016293fc851978
Status: Downloaded newer image for alpine:3.18.4
docker.io/library/alpine:3.18.4

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview alpine:3.18.4

C:\Users\USER>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
/ # apk update
fetch https://dl-cdn.alpinelinux.org/alpine/v3.18/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.18/community/x86_64/APKINDEX.tar.gz
v3.18.6-177-g0fd8a8ca1b2 [https://dl-cdn.alpinelinux.org/alpine/v3.18/main]
v3.18.6-177-g0fd8a8ca1b2 [https://dl-cdn.alpinelinux.org/alpine/v3.18/community]
OK: 20079 distinct packages available
/ # apk upgrade
Upgrading critical system libraries and apk-tools:
(1/1) Upgrading apk-tools (2.14.0-r2 -> 2.14.4-r0)
Executing busybox-1.36.1-r2.trigger
Continuing the upgrade transaction with new apk-tools:
fetch https://dl-cdn.alpinelinux.org/alpine/v3.18/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.18/community/x86_64/APKINDEX.tar.gz
(1/8) Upgrading musl (1.2.4-r1 -> 1.2.4-r2)
(2/8) Upgrading busybox (1.36.1-r2 -> 1.36.1-r5)
Executing busybox-1.36.1-r5.post-upgrade
(3/8) Upgrading busybox-binsh (1.36.1-r2 -> 1.36.1-r5)
(4/8) Upgrading ca-certificates-bundle (20230506-r0 -> 20240226-r0)
(5/8) Upgrading libcrypto3 (3.1.3-r0 -> 3.1.4-r6)
(6/8) Upgrading libssl3 (3.1.3-r0 -> 3.1.4-r6)
(7/8) Upgrading ssl_client (1.36.1-r2 -> 1.36.1-r5)
(8/8) Upgrading musl-utils (1.2.4-r1 -> 1.2.4-r2)
Executing busybox-1.36.1-r5.trigger
OK: 7 MiB in 15 packages
/ # apk add curl
(1/7) Installing ca-certificates (20240226-r0)
(2/7) Installing brotli-libs (1.0.9-r14)
(3/7) Installing libunistring (1.1-r1)
(4/7) Installing libidn2 (2.3.4-r1)
(5/7) Installing nghttp2-libs (1.57.0-r0)
(6/7) Installing libcurl (8.5.0-r0)
(7/7) Installing curl (8.5.0-r0)
Executing busybox-1.36.1-r5.trigger
Executing ca-certificates-20240226-r0.trigger
OK: 12 MiB in 22 packages
```

Lo que sigue es instalar y ejecutar la imagen Nginx con los siguientes comandos

`docker pull nginx:1.23`

`docker run -d nginx:1.23`

```

C:\> Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.4291]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\USER>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS        NAMES
C:\Users\USER>docker run -d nginx:1.23
Unable to find image 'nginx:1.23' locally
1.23: Pulling from library/nginx
f03b40093957: Pull complete
0972072e0e8a: Pull complete
a85095acb896: Pull complete
d24b987aa74e: Pull complete
6c1a86118ade: Pull complete
9989f7b33228: Pull complete
Digest: sha256:f5747a42e3adcb3168049d63278d7251d91185bb5111d2563d58729a5c9179b0
Status: Downloaded newer image for nginx:1.23
9741d8b88a8f6a6c1cf7b4115e95c1388793e8589740bd4592fa709a9b6cec00

C:\Users\USER>
```

Verificamos que el contenedor esté en ejecución

`docker ps`

Despues Exponemos un puerto y acceder al servidor Nginx:

Ejecutamos el siguiente comando para exponer el puerto 9090 del contenedor Nginx al puerto 80 del host:

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

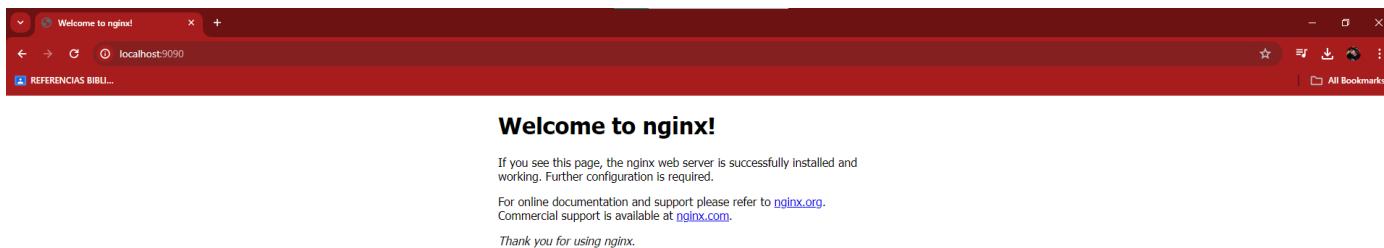
```
C:\ Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.4291]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\USER>docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS        NAMES
C:\Users\USER>docker run -d nginx:1.23
Unable to find image 'nginx:1.23' locally
1.23: Pulling from library/nginx
f03b40093957: Pull complete
0972072e0e8a: Pull complete
a85095acb896: Pull complete
d24b987aa74e: Pull complete
6c1a86118ade: Pull complete
9989f7b33228: Pull complete
Digest: sha256:f5747a42e3adcb3168049d63278d7251d91185bb5111d2563d58729a5c9179b0
Status: Downloaded newer image for nginx:1.23
9741d8b88a8f6a6c1cf7b4115e95c1388793e8589740bd4592fa709a9b6cec00

C:\Users\USER>docker run -d -p 9090:80 nginx:1.23
8e1b70628858caadb7870d3b766a984c843a512a588918004a664a626fa92077

C:\Users\USER>
```

Y Abriremos en el navegador web y visita localhost:9090 para ver el mensaje de bienvenida de Nginx.



Despues le daremos el nombre al contenedor:

Ejecutamos el siguiente comando para asignar un nombre al contenedor

```
docker run --name mi-web-app -d -p 9090:80 nginx:1.23
```

Crearemos una aplicación Node.js

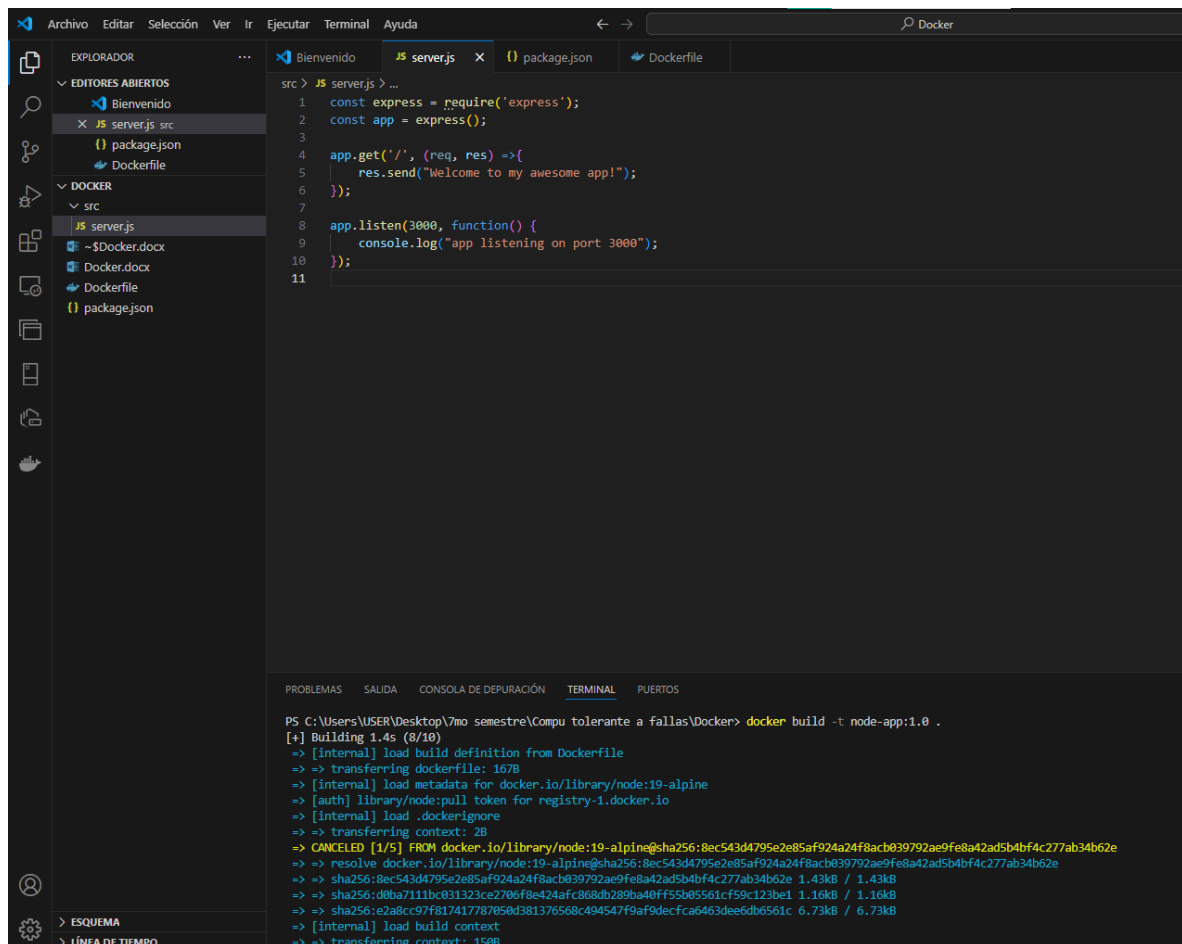
Abriremos el editor de texto (como Visual Studio Code) y crea el siguiente archivo src/server.js con el siguiente contenido:

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

const app = express();

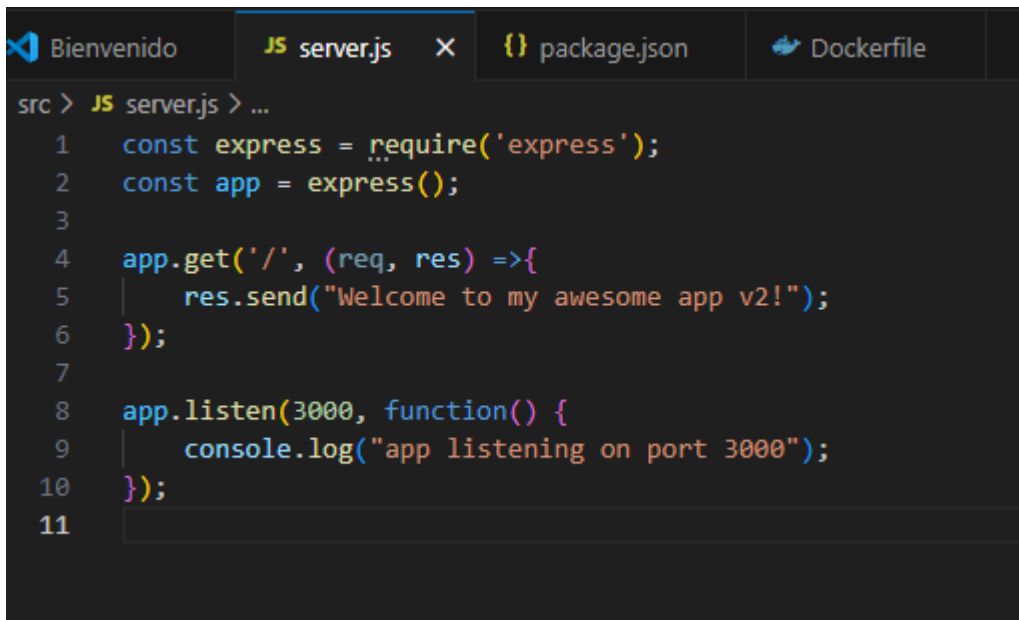
app.get('/', (req, res) => {
  res.send("Welcome to my awesome app!");
});

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



Y creamos con el siguiente comando un archivo package.json en la raíz del proyecto con el siguiente contenido

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

A screenshot of a code editor with a dark theme. The editor has four tabs at the top: 'Bienvenido', 'JS server.js', 'package.json', and 'Dockerfile'. The 'JS server.js' tab is active. The code in the editor is as follows:

```
src > JS server.js > ...  
1  const express = require('express');  
2  const app = express();  
3  
4  app.get('/', (req, res) =>{  
5    res.send("Welcome to my awesome app v2!");  
6  });  
7  
8  app.listen(3000, function() {  
9    console.log("app listening on port 3000");  
10 });  
11
```

Despues crearemos el Dockerfile para la aplicación Node.js:

Creamos un archivo Dockerfile en la raíz del proyecto con el siguiente contenido

FROM node:19-alpine

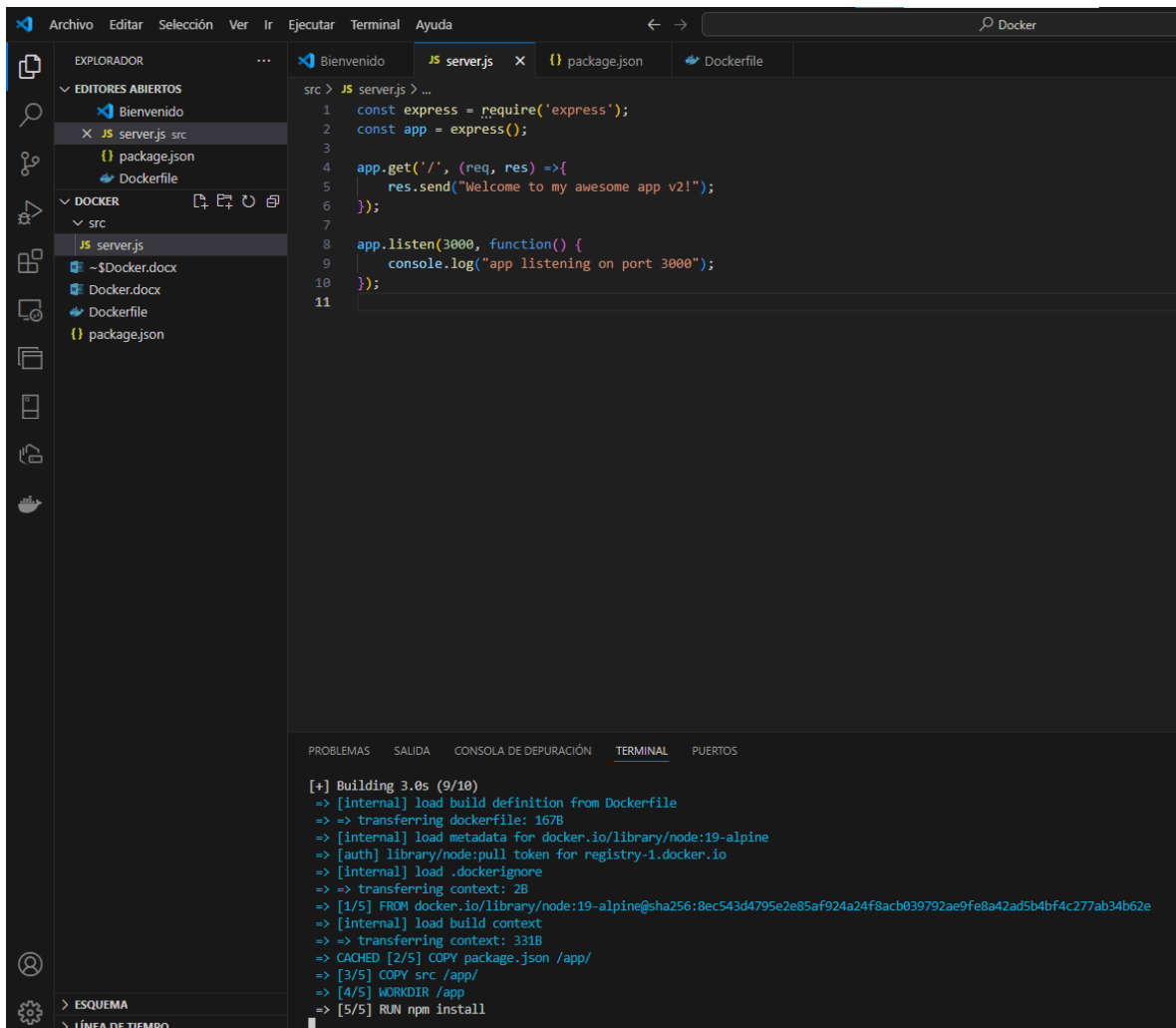
COPY package.json /app/

COPY src /app/

WORKDIR /app

RUN npm install

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



Construiremos y ejecutaremos la imagen de la aplicación Node.js:

En la terminal, navegamos hasta la raíz del proyecto y ejecutamos el siguiente comando para construir la imagen de la aplicación Node.js:

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

Ejecuta un contenedor basado en esta imagen:

```
docker run -d -p 3000:3000 node-app:1.0
```

Y a continuación podremos ver el mensaje de bienvenida de nuestra aplicación



Por último, modificamos la aplicación y actualizaremos la imagen Docker:

Modificamos el archivo `src/server.js` para cambiar el mensaje de bienvenida.

Detenemos el contenedor existente utilizando su ID o nombre.

Reconstruimos la imagen de la aplicación Node.js.

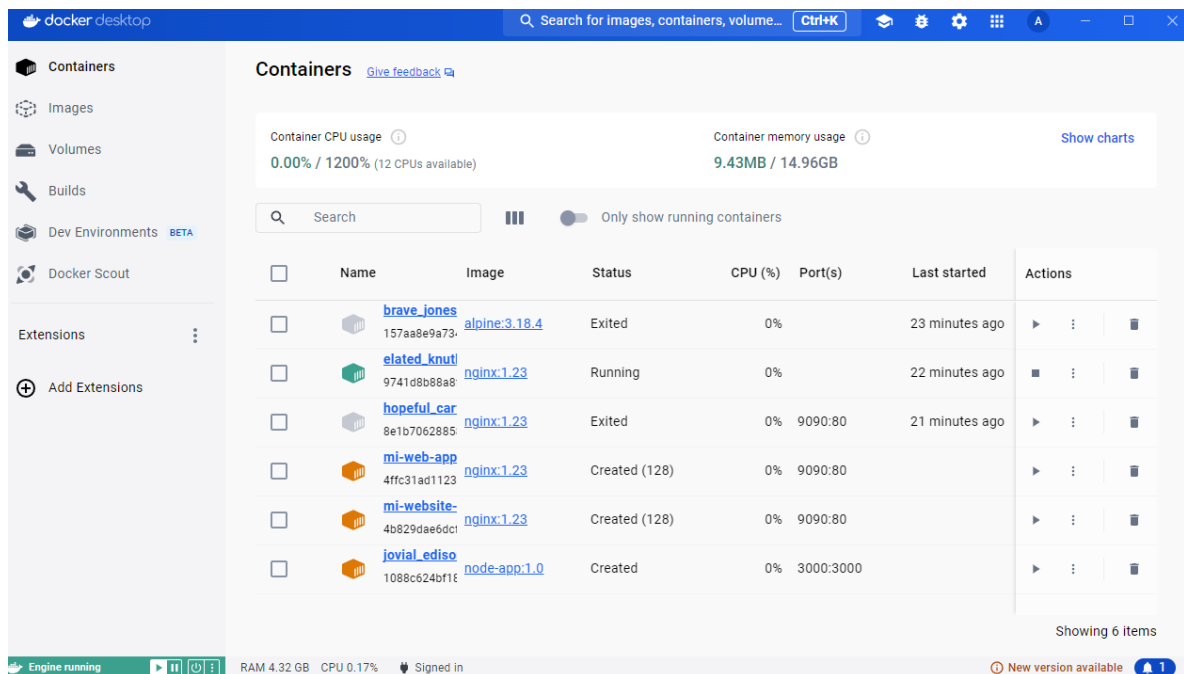
Ejecutamos un nuevo contenedor basado en la versión actualizada de la imagen.

Algo que tuve que hacer porque tuve un conflicto fue Cambiar el puerto de mi aplicación Node.js:

Ya que no puedes liberar el puerto 4000 por que en el anterior también tuve este error en mi sistema, intenté cambiar el puerto en el que mi aplicación Node.js escucha. Modifique el archivo `src/server.js` para que la aplicación escuche en un puerto diferente, por ejemplo, el puerto 5000.



Aquí tenemos la aplicación Docker desktop y vemos los cambios que se fueron realizando y lo que estuve haciendo.



En conclusión, trabajar con Docker y modificar una aplicación Node.js ha sido una experiencia enriquecedora que me ha permitido explorar el mundo de la virtualización de contenedores y el desarrollo de aplicaciones. A través de este proceso, he comprendido la importancia de Docker en la creación, distribución y despliegue de aplicaciones de manera consistente y eficiente. He tenido la oportunidad de enfrentar desafíos como conflictos de puertos y errores de configuración, lo que me ha permitido desarrollar mis habilidades de resolución de problemas y aprender a utilizar herramientas de diagnóstico como **netstat** y **docker logs**. Además, al modificar la aplicación Node.js, he fortalecido mis habilidades en el desarrollo de aplicaciones web utilizando Node.js y Express. En resumen, esta experiencia me ha brindado un sólido entendimiento de Docker y su integración con aplicaciones Node.js, lo que seguramente será invaluable en mi trayectoria profesional en el desarrollo de software.