



# Práctica (obligatoria)

## Despliegue aplicación de gestión de órdenes de compra de un almacén

Despliegue la aplicación Nodejs desarrollada durante el módulo 1 en la parte de microservicios (Gestión de las órdenes de compra de un almacen) en Docker de acuerdo a los siguientes pasos:

1. Describa sus componentes y arquitectura
2. Despléguela, haciendo uso de tecnologías como Docker, Docker Compose, Docker Swarm
3. Realice pruebas de escalabilidad y de carga con JMeter

### ▼ Recomendaciones

- Establezca las relaciones entre los microservicios desplegados y revise que su docker-compose.yml refleje dichas relaciones.
- Revise que los llamados a los microservicios desde otros usen los nombres de los servicios especificados en el docker-compose.yml
- Puede usar imágenes oficiales de dockerhub con paquetes preinstalados tales como:

node ([https://hub.docker.com/\\_/node](https://hub.docker.com/_/node))

php ( [https://hub.docker.com/\\_/php](https://hub.docker.com/_/php)), esta tiene un tag específico para trabajar con apache.

- En la aplicación web php, se recomienda mover el bloque de código php que inicia la sesión, al inicio de los archivos .php para evitar warnings de

envío de headers. Adicionalmente, no dejar ningún espacio en el archivo antes de la etiqueta `<?php`. Al igual que evitar realizar "echos" antes de los envíos de cabecera (`header()`)

---

## ▼ Componentes y arquitectura

### ▼ Relaciones entre los microservicios

#### **USUARIOS**

Administrar los datos de los usuarios y verificar sus credenciales

#### **PRODUCTOS**

Administrar la información sobre los productos y reducir el inventario cuando se realicen compras.

#### **ORDENES**

Recibir los detalles de los productos que los usuarios desean comprar, verificar su disponibilidad, calcular el costo total y crear la orden correspondiente.

---

La relación entre las entidades en una arquitectura de microservicios se establece de la siguiente manera:

- **ORDENES y PRODUCTOS:** Cada vez que se reciba una orden en **ORDENES**, se verificará la disponibilidad de los productos en **PRODUCTOS** y se obtendrá el precio para calcular el total de la orden.
- **ORDENES y USUARIOS:** Al crear una orden en **ORDENES**, se solicitarán los datos de nombre y correo electrónico al usuario a **USUARIOS** para agregarlos a la orden de compra.

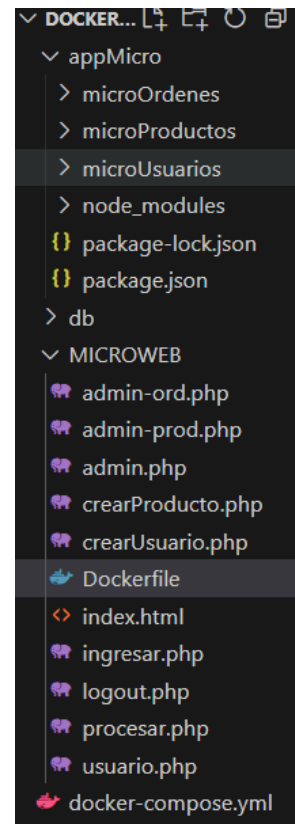
---

## ▼ Estructura de carpetas

- **db** tiene el script de MySQL
- **appMicro:**
  - microUsuarios → 3001
  - microProductos → 3002
  - microOrdenes → 3003
- **MICROWEB** tiene los archivos PHP y HTML para la página web



En cada microservicio está **src** donde se encuentran los **controller**, **models** y su **index.js**



## ▼ docker-compose.yml

### ▼ 5 servicios

- **Web** → 8080:80
- **MySQL (db)** → 3306
- **microUsuarios** → 3001
- **microProductos** → 3002
- **microOrdenes** → 3003

```
services:
  db:
```

```
image: mysql:5.7
container_name: db
ports:
  - "32000:3306"
environment:
  - MYSQL_ROOT_PASSWORD=password
  - MYSQL_DATABASE=almacen
volumes:
  - ./db:/docker-entrypoint-initdb.d
usuarios:
  build: ./appMicro/microUsuarios
  container_name: usuarios
  ports:
    - "3001:3001"
  links:
    - db
productos:
  build: ./appMicro/microProductos
  container_name: productos
  ports:
    - "3002:3002"
  links:
    - db
ordenes:
  build: ./appMicro/microPrdenes
  container_name: ordenes
  ports:
    - "3003:3003"
  links:
    - db
    - usuarios
    - productos
web:
  build: ./web
  container_name: web
  ports:
```

```
- "8080:80"
links:
  - usuarios
  - productos
  - ordenes
```

## ▼ MySQL

```
DROP TABLE IF EXISTS usuarios;
CREATE TABLE usuarios (
  nombre varchar(50),
  email varchar(50),
  usuario varchar(50),
  password varchar(50),
  PRIMARY KEY(usuario)
);

DROP TABLE IF EXISTS productos;
CREATE TABLE productos (
  id int(11) NOT NULL AUTO_INCREMENT,
  nombre varchar(50),
  precio decimal(10,2),
  inventario int(11),
  PRIMARY KEY(id)
);

DROP TABLE IF EXISTS ordenes;
CREATE TABLE ordenes (
  id int(11) NOT NULL AUTO_INCREMENT,
  nombreCliente varchar(50),
  emailCliente varchar(50),
  totalCuenta decimal(10,2),
  fecha DATETIME DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY(id)
);
```

```
INSERT INTO usuarios (nombre, email, usuario, password) VALUES ('', '', '', '')
INSERT INTO usuarios (nombre, email, usuario, password) VALUES ('', '', '', '')
```

## ▼ MICROWEB

### ▼ Cambios a los codigos

#### ▼ Paginas de usuario y admin

##### Antes

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" rel="script">
  <title>Document</title>
</head>
<body>
  <?php
    session_start();
    $us=$_SESSION["usuario"];
    if ($us==""){
      header("location: index.html");
    }
  ?>
</body>
```

##### Después

```
<?php
  session_start();
  $us=$_SESSION["usuario"];
  if ($us==""){
    header("Location: index.html");
  }
?>

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js" rel="script">
  <title>Document</title>
</head>
<body>
```

#### ▼ ingresar.php

##### Antes

##### Después

```
<?php
$user=$_POST["usuario"];
$pass=$_POST["password"];
#$servurl="http://localhost:3001/usuarios/$user/$pass";
$servurl="http://192.168.100.2:3001/usuarios/$user/$pass";
$curl=curl_init($servurl);

curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
$response=curl_exec($curl);
curl_close($curl);
if ($response===false){
    header("Location:index.html");
}
$resp = json_decode($response);
if (count($resp) != 0){
    session_start();
    $_SESSION["usuario"]=$user;
    if ($user == "admin"){
        echo "admin";
        header("Location:admin.php");
    }
    else {
        echo "usuario";
        header("Location:usuario.php");
    }
}
else {
    header("Location:index.html");
}
?>
```

```
<?php
$user=$_POST["usuario"];
$pass=$_POST["password"];
#$servurl="http://localhost:3001/usuarios/$user/$pass";
$servurl="http://192.168.100.2:3001/usuarios/$user/$pass";
$curl=curl_init($servurl);

curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
$response=curl_exec($curl);
curl_close($curl);
if ($response===false){
    header("Location:index.html");
}
$resp = json_decode($response);
if (count($resp) != 0){
    session_start();
    $_SESSION["usuario"]=$user;
    if ($user == "admin"){
        //echo "admin";
        header("Location:admin.php");
    }
    else {
        //echo "usuario";
        header("Location:usuario.php");
    }
}
else {
    header("Location:index.html");
}
?>
```

## ▼ procesar.php

Antes

```
if ($SERVER["REQUEST_METHOD"] == "POST") {
    if (isset($_POST["cantidad"])) {
        $ch = curl_init();

        // Configurar opciones de curl
        curl_setopt($ch, CURLOPT_URL, $url);
        curl_setopt($ch, CURLOPT_POST, true);
        curl_setopt($ch, CURLOPT_POSTFIELDS, $json);
        curl_setopt($ch, CURLOPT_HTTPHEADER, array('content-type: application/json'));
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

        // Ejecutar la solicitud POST
        $response = curl_exec($ch);

        // Verificar si hubo algún error en la solicitud
        if ($response === false) {
            header("location: index.html");
        }

        // Cerrar la conexión curl
        curl_close($ch);

        // Mostrar mensaje de éxito
        echo "La orden ha sido creada correctamente.";
        header("location: usuario.php");
    }
    else {
        echo "Por favor, seleccione al menos un producto y especifique la cantidad.";
    }
}
else {
    header("location: index.html");
    exit();
}
?>
```

Después

```
if ($SERVER["REQUEST_METHOD"] == "POST") {
    if (isset($_POST["cantidad"])) {
        $ch = curl_init();

        // Configurar opciones de curl
        curl_setopt($ch, CURLOPT_URL, $url);
        curl_setopt($ch, CURLOPT_POST, true);
        curl_setopt($ch, CURLOPT_POSTFIELDS, $json);
        curl_setopt($ch, CURLOPT_HTTPHEADER, array('content-type: application/json'));
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

        // Ejecutar la solicitud POST
        $response = curl_exec($ch);

        // Verificar si hubo algún error en la solicitud
        if ($response === false) {
            header("location: index.html");
        }

        // Cerrar la conexión curl
        curl_close($ch);

        // Mostrar mensaje de éxito
        echo "La orden ha sido creada correctamente.";
        header("location: usuario.php");
    }
    else {
        echo "Por favor, seleccione al menos un producto y especifique la cantidad.";
    }
}
else {
    header("location: index.html");
    exit();
}
?>
```

## ▼ Los nombres de los servicios (URLs)

En vez de colocar la IP del servidor o *localhost*, se debe colocar el nombre del servicio. Recordar que las IPs de los contenedores pueden variar, por tanto nos aseguramos de realizar estos cambios. El ejemplo de la imagen aplica a las otras URLs de los otros archivos.

```
<?php
$user=$_POST["usuario"];
$pass=$_POST["password"];
$servurl="http://microUsuarios:3001/usuarios/$user/$pass";
$curl=curl_init($servurl);

curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
$response=curl_exec($curl);
curl_close($curl);
```

## ▼ Dockerfile

```
FROM php:8.1-apache

COPY . /var/www/html
```

## ▼ Microservicios

### ▼ npm start



Esto es opcional, se puede evitar todo este proceso simplemente cambiando `npm start` por `node src/index.js` en el Dockerfile de cada microservicio.

Se usa npm start para ejecutar los microservicios. Para que este comando se ejecute normalmente hay que agregar en package.json, exactamente en el apartado de scripts, un script para "start" que sea igual a "node src/index". Este proceso se hace para cada uno de los package.json de los microservicios.



```

appMicro > microOrdenes > {} package.json > {} dependencies
1  {
2    "name": "microordenes",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "start": "node src/index.js"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "axios": "^1.6.7",
14     "express": "^4.18.3",
15     "morgan": "^1.10.0",
16     "mysql": "^2.18.1",
17     "mysql2": "^3.9.2"
18   }
19 }
20

```

## ▼ microUsuarios

### ▼ Dockerfile

```

FROM node:20

WORKDIR /usuarios
COPY package.json .
RUN npm install

EXPOSE 3001

COPY . .
CMD npm start

```

## ▼ microProductos

### ▼ Dockerfile

```

FROM node:20

WORKDIR /productos

```

```
COPY package.json .  
RUN npm install
```

```
EXPOSE 3002
```

```
COPY . .  
CMD npm start
```

## ▼ microOrdenes

### ▼ Dockerfile

```
FROM node:20
```

```
WORKDIR /ordenes  
COPY package.json .  
RUN npm install
```

```
EXPOSE 3003
```

```
COPY . .  
CMD npm start
```

---

## ▼ Despliegue

| *Los archivos se pasaron a la máquina mediante `git clone`*

```
git clone https://github.com/SEBASBELMOS/DOCKER-ALMACEN.git
```

```
root@servidorUbuntu:~# git clone https://github.com/SEBASBELMOS/DOCKER-ALMACEN.git  
Cloning into 'DOCKER-ALMACEN'...  
remote: Enumerating objects: 1245, done.
```

## ▼ Mediante Docker Compose

[illegible]

```

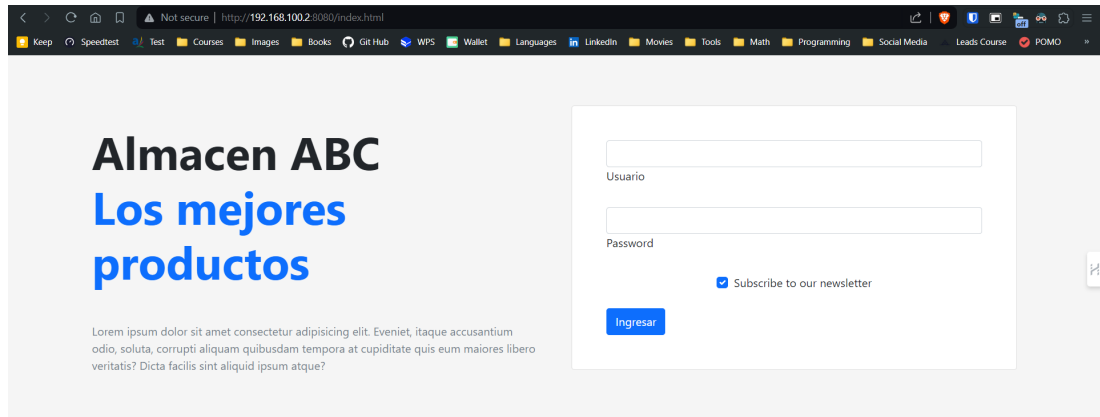
=> sha256:7085e84b1dfb39125139645f45c3b415ca9dd9ca37e85033cf0f1fb1d308760 12.18MB / 12.18MB 40.2s
=> sha256:2348e0c590e6dd6789d8e29bdf24524707fbc96c1597dafde4834b4eb8862 493B / 493B 41.3s
=> sha256:ab0f3d17ba76ff837f2b310ba550943ad644fa5362542691235d8b35e953922e 11.16MB / 11.16MB 47.4s
=> sha256:a7065a1437aa90679237742f587bb04c3c6063d1dd20f5c910587a362f9411b4 2.46kB / 2.46kB 44.1s
=> sha256:d2da8d5f04265538298965d2a1168aa6c38ae3624f797a27c6f735bee6d328d5 244B / 244B 44.4s
=> sha256:c645d761a45c1471383509f875794c7542ac1fb4edf93d179ed04733308a14e 894B / 894B 44.7s
=> extracting sha256:ff65b997523ed8926becf7a45f5beab5cc1741de8c518e8f7e62942018c 6.6s
=> extracting sha256:46d87a00aaae1ae22413715f1cacb07a29aa3dfbf4d90561b2966e73bd2efbc8 0.0s
=> extracting sha256:6b61c519885f76f9cad67cb71136fd446b3b573c76c466114feb5261c0f97cd 1.1s
=> extracting sha256:4833ee2f6ebecf460037afa49512ad9000488709c6642663fc48cea6a04f9c2c3 0.0s
=> extracting sha256:f4e4debe797ac502171678858265da37cd544b7b50de252d82e9b6f7733364a1 0.0s
=> extracting sha256:7085e84b1dfb39125139645f45c3b415ca9dd9ca37e85033cf0f1fb1d308760 0.2s
=> sha256:2348e0c590e6dd6789d8e29bdf24524707fbc96c1597dafde4834b4eb8862 894B 0.0s
=> extracting sha256:ab0f3d17ba76ff837f2b310ba550943ad644fa5362542691235d8b35e953922e 0.7s
=> sha256:a7065a1437aa90679237742f587bb04c3c6063d1dd20f5c910587a362f9411b4 0.0s
=> extracting sha256:d2da8d5f04265538298965d2a1168aa6c38ae3624f797a27c6f735bee6d328d5 0.0s
=> extracting sha256:c645d761a45c1471383509f875794c7542ac1fb4edf93d179ed04733308a14e 0.0s
=> [web 2/2] COPY . /var/www/html 0.4s
=> [web] exporting to image 0.1s
=> exporting layers 0.0s
=> writing image sha256:d7b6f230022575ac5f9002e67fbfb874dd1e4f621c49637bcef55a9dc72fe769 0.0s
=> naming to docker.io/library/docker-almacen-web 0.0s

[+] Running 5/6
✔ Network docker-almacen.default Created 2.0s
✔ Container db Started 0.6s
✔ Container productos Started 0.9s
✔ Container usuarios Started 0.9s
✔ Container ordenes Started 1.2s
✔ Container web Started 1.7s

root@seaviden01buntu:~/DOCKER-ALMACEN# |

```

Práctica (obligatoria) Despliegue aplicación de gestión de órdenes de compra de un almacén



## ▼ Mediante Docker Swarm

### ▼ Cambiar el `docker-compose.yml`

#### ▼ Reemplazar lo siguiente

- Eliminar los `container_name`
- Reemplazar `links`  
→ `depends_on`
- Comentar `build` y crear una imagen para cada servicio
- Crear `networks`

```
networks:|
  cluster_almacenrei_default:
services:
  db:
    image: mysql:5.7
    ports:
      - "32000:3306"
    environment:
      - MYSQL_ROOT_PASSWORD=password
      - MYSQL_DATABASE=almacen
    volumes:
      - ./db:/docker-entrypoint-initdb.d
  usuarios:
    #build: ./appMicro/microUsuarios
    image: sebasbelmos/usuarios_almacen
    ports:
      - "3001:3001"
    depends_on:
      - db
  productos:
    #build: ./appMicro/microProductos
    image: sebasbelmos/productos_almacen
    ports:
      - "3002:3002"
    depends_on:
      - db
  ordenes:
    #build: ./appMicro/microOrdenes
    image: sebasbelmos/ordenes_almacen
    ports:
      - "3003:3003"
    depends_on:
      - db
      - usuarios
      - productos
  web:
```

```
networks:
  cluster_almacerei_default:
services:
```

```
db:
  image: mysql:5.7
  ports:
    - "32000:3306"
  environment:
    - MYSQL_ROOT_PASSWORD=password
    - MYSQL_DATABASE=almacen
  volumes:
    - ./db:/docker-entrypoint-initdb.d
usuarios:
  image: sebasbelmos/usuarios_almacen
  ports:
    - "3001:3001"
  links:
    - db
productos:
  image: sebasbelmos/productos_almacen
  ports:
    - "3002:3002"
  links:
    - db
ordenes:
  image: sebasbelmos/ordenes_almacen
  ports:
    - "3003:3003"
  links:
    - db
    - usuarios
    - productos
web:
  image: sebasbelmos/web_almacen
  ports:
    - "8080:80"
  links:
    - usuarios
```

- productos
- ordenes

▼ Iniciar el clúster con: `docker swarm init --advertise-addr 192.168.100.2`

```
root@servidorUbuntu:~/DOCKER-ALMACEN# docker swarm init
Swarm initialized: current node (wprmnfugnrdu2aibwzjeu
```

To add a worker to this swarm, run the following command:

```
docker swarm join --token SWMTKN-1-29b3hfueu1ft6fs1
```

To add a manager to this swarm, run 'docker swarm join-t

▼ Construir las imágenes y subirlas a Docker Hub

▼ Web

```
docker build -t sebasbelmos/web_almacen .
```

```
root@servidorUbuntu:~/DOCKER-ALMACEN/MICROWEB# docker
[+] Building 0.0s (0/0)  docker:default
[+] Building 1.3s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 80B
=> [internal] load metadata for docker.io/library/pl
=> [auth] library/php:pull token for registry-1.docl
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 26.95kB
=> CACHED [1/2] FROM docker.io/library/php:8.1-apac
=> [2/2] COPY . /var/www/html
=> exporting to image
=> => exporting layers
```

```
=> => writing image sha256:0214d142744c856f88eaa4c3f  
=> => naming to docker.io/sebasbelmos/web_almacen
```

```
docker push sebasbelmos/web_almacen
```

```
root@servidorUbuntu:~/DOCKER-ALMACEN/MICROWEB# docker  
Using default tag: latest  
The push refers to repository [docker.io/sebasbelmos,  
87f9dcc9a5d5: Pushed  
2f9fd4afcebb: Mounted from library/php  
db35f924b59c: Mounted from library/php  
00230e47d0c9: Mounted from library/php  
c24f383c7cf6: Mounted from library/php  
3a0ffcd3b552: Mounted from library/php  
e7c24cbffe5a: Mounted from library/php  
4b713f2e5bc0: Mounted from library/php  
8eedaabd7bf7: Mounted from library/php  
05769640ee18: Mounted from library/php  
ee50ac85ebd7: Mounted from library/php  
781dffa2223b: Mounted from library/php  
1706a4264cc9: Mounted from library/php  
1f00ff201478: Mounted from library/php  
latest: digest: sha256:e7ef4158dbaec0c1e087ee512853c
```

## ▼ Microservicios

### ▼ Usuarios

```
docker build -t sebasbelmos/usuarios_almacen .
```

```
root@servidorUbuntu:~/DOCKER-ALMACEN/appMicro/micro  
[+] Building 0.0s (0/1)  
[+] Building 53.8s (11/11) FINISHED  
=> [internal] load build definition from Dockerfile
```

```
=> => transferring dockerfile: 155B
=> [internal] load metadata for docker.io/library/
=> [auth] library/node:pull token for registry-1
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/node:20@sha256:84
=> [internal] load build context
=> => transferring context: 7.42MB
=> CACHED [2/5] WORKDIR /microUsuarios
=> CACHED [3/5] COPY package.json .
=> CACHED [4/5] RUN npm install
=> [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:e4d5fe8aa1e715da35ea61
=> => naming to docker.io/sebasbelmos/usuarios_a
```

```
docker push sebasbelmos/usuarios_almacen
```

```
root@servidorUbuntu:~/DOCKER-ALMACEN/appMicro/microUsuarios
Using default tag: latest
The push refers to repository [docker.io/sebasbelmos/usuarios_almacen]
a6292d6d3aff: Pushed
ea6105dcd1b5: Pushed
a35732378f36: Pushed
e4789fd2af61: Pushed
123194e8ca81: Mounted from library/node
a28e61f75fa0: Mounted from library/node
41c2d1f0a1d3: Mounted from library/node
3e81cc85b636: Mounted from library/node
893507f6057f: Mounted from library/node
2353f7120e0e: Mounted from library/node
51a9318e6edf: Mounted from library/node
```



```
c5bb35826823: Mounted from library/node
latest: digest: sha256:ccfd37d444d59aad91debfd6006
```

## ▼ Productos

```
docker build -t sebasbelmos/productos_almacen .
```

```
root@servidorUbuntu:~/DOCKER-ALMACEN/appMicro/microProductos
[+] Building 8.3s (4/9)
[+] Building 8.9s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 156B
=> [internal] load metadata for docker.io/library/node:20@sha256:84
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/node:20@sha256:84
=> [internal] load build context
=> => transferring context: 7.42MB
=> CACHED [2/5] WORKDIR /microProductos
=> CACHED [3/5] COPY package.json .
=> CACHED [4/5] RUN npm install
=> [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:eda628e7f829650b44213c
=> => naming to docker.io/sebasbelmos/productos_almacen
```

```
docker push sebasbelmos/productos_almacen
```

```
root@servidorUbuntu:~/DOCKER-ALMACEN/appMicro/microProductos
Using default tag: latest
The push refers to repository [docker.io/sebasbelmos/productos_almacen]
d05214c75006: Pushed
```

```
502f9efc7c02: Pushed
72e570e7ea7d: Pushed
9a06e570d26f: Pushed
123194e8ca81: Mounted from sebasbelmos/usuarios_a
a28e61f75fa0: Mounted from sebasbelmos/usuarios_a
41c2d1f0a1d3: Mounted from sebasbelmos/usuarios_a
3e81cc85b636: Mounted from sebasbelmos/usuarios_a
893507f6057f: Mounted from sebasbelmos/usuarios_a
2353f7120e0e: Mounted from sebasbelmos/usuarios_a
51a9318e6edf: Mounted from sebasbelmos/usuarios_a
c5bb35826823: Mounted from sebasbelmos/usuarios_a
latest: digest: sha256:582968cdc6087ef4d22206e32dl
```

#### ▼ Ordenes

```
docker build -t sebasbelmos/ordenes_almacen .
```

```
root@servidorUbuntu:~/DOCKER-ALMACEN/appMicro/microOrdenes
[+] Building 1.3s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 154B
=> [internal] load metadata for docker.io/library/node:20@sha256:84
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/node:20@sha256:84
=> [internal] load build context
=> => transferring context: 7.42MB
=> CACHED [2/5] WORKDIR /microOrdenes
=> CACHED [3/5] COPY package.json .
=> CACHED [4/5] RUN npm install
=> [5/5] COPY . .
=> exporting to image
=> => exporting layers
```

```
=> => writing image sha256:2c7b70b638360a49398d98
=> => naming to docker.io/sebasbelmos/ordenes_almacen
```

```
docker push sebasbelmos/ordenes_almacen
```

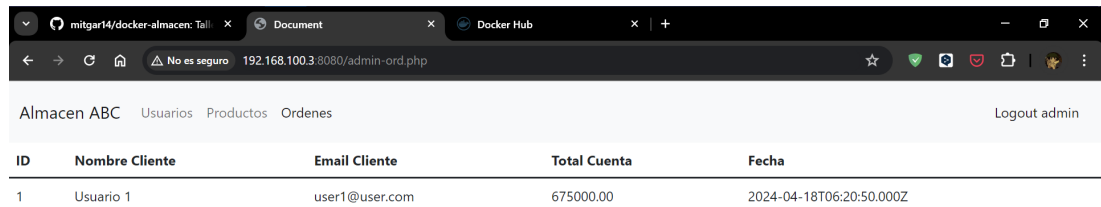
```
root@servidorUbuntu:~/DOCKER-ALMACEN/appMicro/micro
Using default tag: latest
The push refers to repository [docker.io/sebasbelmos/ordenes_almacen]
2e9fe9937029: Pushed
4f59b4699806: Pushed
302283928095: Pushed
859a5e8cb562: Pushed
123194e8ca81: Mounted from sebasbelmos/productos_almacen
a28e61f75fa0: Mounted from sebasbelmos/productos_almacen
41c2d1f0a1d3: Mounted from sebasbelmos/productos_almacen
3e81cc85b636: Mounted from sebasbelmos/productos_almacen
893507f6057f: Mounted from sebasbelmos/productos_almacen
2353f7120e0e: Mounted from sebasbelmos/productos_almacen
51a9318e6edf: Mounted from sebasbelmos/productos_almacen
c5bb35826823: Mounted from sebasbelmos/productos_almacen
latest: digest: sha256:4b80acecaae2d9ef40dc720df40
```

▼ Ejecutar el stack con `docker stack deploy -c docker-compose.yml stack_almacenrei`

```
root@servidorUbuntu:~/DOCKER-ALMACEN# docker stack deploy
Since --detach=false was not specified, tasks will be created
In a future release, --detach=false will become the default
Creating network stack_almacenrei_default
Creating service stack_almacenrei_web
Creating service stack_almacenrei_db
Creating service stack_almacenrei_usuarios
Creating service stack_almacenrei_productos
Creating service stack_almacenrei_ordenes
```

```
root@servidorUbuntu:~/DOCKER-ALMACEN# docker service ls
ID                NAME                                MODE
5tvb22m91nq2     stack_almacenrei_db               replicated
y59ojr7oe5w      stack_almacenrei_ordenes          replicated
09kvqsadpl0a     stack_almacenrei_productos        replicated
werewecqp1k2     stack_almacenrei_usuarios         replicated
awpya8ykzy5c     stack_almacenrei_web              replicated
```

### ▼ Verificar el funcionamiento



ID	Nombre Cliente	Email Cliente	Total Cuenta	Fecha
1	Usuario 1	user1@user.com	675000.00	2024-04-18T06:20:50.000Z

### ▼ Escalar el servicio web

```
docker service scale stack_almacen_web=6
docker service ls
docker service ps stack_almacen_web
```

```
root@servidorUbuntu:~/cluster-almacen# docker service ls
ID                NAME                MODE                REPLICAS        IMAGE                PORTS
ulyfa0v6uxy0     stack_almacen_db     replicated           1/1             mysql:5.7           *:32000->3306/tcp
rhckp7y3dlb8     stack_almacen_ordenes replicated           1/1             mitgar14/ordenes-almacen:latest *:3003->3003/tcp
quqcnp9fhr50     stack_almacen_productos replicated           1/1             mitgar14/productos-almacen:latest *:3002->3002/tcp
t0sf8yws2vs      stack_almacen_usuarios replicated           1/1             mitgar14/usuarios-almacen:latest *:3001->3001/tcp
ljfbjpwitl8v     stack_almacen_web     replicated           1/1             mitgar14/web-almacen:latest      *:8080->80/tcp

root@servidorUbuntu:~/cluster-almacen# docker service scale stack_almacen_web=6
docker service ls
stack_almacen_web scaled to 6
overall progress: 6 out of 6 tasks
1/6: running [=====]
2/6: running [=====]
3/6: running [=====]
4/6: running [=====]
5/6: running [=====]
6/6: running [=====]
verify: Service stack_almacen_web converged
ID                NAME                MODE                REPLICAS        IMAGE                PORTS
ulyfa0v6uxy0     stack_almacen_db     replicated           1/1             mysql:5.7           *:32000->3306/tcp
rhckp7y3dlb8     stack_almacen_ordenes replicated           1/1             mitgar14/ordenes-almacen:latest *:3003->3003/tcp
quqcnp9fhr50     stack_almacen_productos replicated           1/1             mitgar14/productos-almacen:latest *:3002->3002/tcp
t0sf8yws2vs      stack_almacen_usuarios replicated           1/1             mitgar14/usuarios-almacen:latest *:3001->3001/tcp
ljfbjpwitl8v     stack_almacen_web     replicated           6/6             mitgar14/web-almacen:latest      *:8080->80/tcp

ID                NAME                IMAGE                NODE                DESIRED STATE        CURRENT STATE        ERROR                PORTS
28gss2vduerq     stack_almacen_web.1  mitgar14/web-almacen:latest  servidorUbuntu     Running               Running 23 seconds ago
3fyo0wmt9p9      _\ stack_almacen_web.1  mitgar14/web-almacen:latest  servidorUbuntu     Shutdown              Shutdown 27 seconds ago
6ft8py6jovo8     _\ stack_almacen_web.1  mitgar14/web-almacen:latest  servidorUbuntu     Shutdown              Shutdown about a minute ago
u1nrbkpmh6jw     _\ stack_almacen_web.1  mitgar14/web-almacen:latest  servidorUbuntu     Shutdown              Shutdown 4 minutes ago
2pleeeydju2q     stack_almacen_web.2  mitgar14/web-almacen:latest  servidorUbuntu     Running               Running 5 seconds ago
root@servidorUbuntu:~/cluster-almacen# stack_almacen_web.3
ID                NAME                IMAGE                NODE                DESIRED STATE        CURRENT STATE        ERROR                PORTS
xmr52yh42z0i     stack_almacen_web.3  mitgar14/web-almacen:latest  servidorUbuntu     Running               Running 5 seconds ago
vdf9ek1ld6ov     stack_almacen_web.4  mitgar14/web-almacen:latest  servidorUbuntu     Running               Running 5 seconds ago
un6n12t6mtga     stack_almacen_web.5  mitgar14/web-almacen:latest  servidorUbuntu     Running               Running 5 seconds ago
xtp5rxybaqn      stack_almacen_web.6  mitgar14/web-almacen:latest  servidorUbuntu     Running               Running 5 seconds ago
root@servidorUbuntu:~/cluster-almacen#
```

## ▼ Pruebas de escalabilidad y carga

| Con 1000 threads y 2 loops

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	2000	511	2	1703	538.22	0.00%	784.0/sec	2681.62	94.17	3502.5
TOTAL	2000	511	2	1703	538.22	0.00%	784.0/sec	2681.62	94.17	3502.5