



MANUAL DE USUARIO

Contenedores para apps Cliente–Servidor (CRUD)

Universidad de San Carlos de Guatemala

Facultad de ingeniería

Escuela de Ciencias y Sistemas

Grupo: Grupo #8

Integrantes:

- Emily Maritza Tepeu Guacamaya
- Jackeline Stephany Rivera Argueta
- Armando Jose Vásquez Castillo
- Gervin de Jesús Del Cid Jolón
- Juan Manuel de León Martínez
- Juan Pablo Morales Salazar
- **Tutores: Juan Samayoa, Enner Mendizabal**

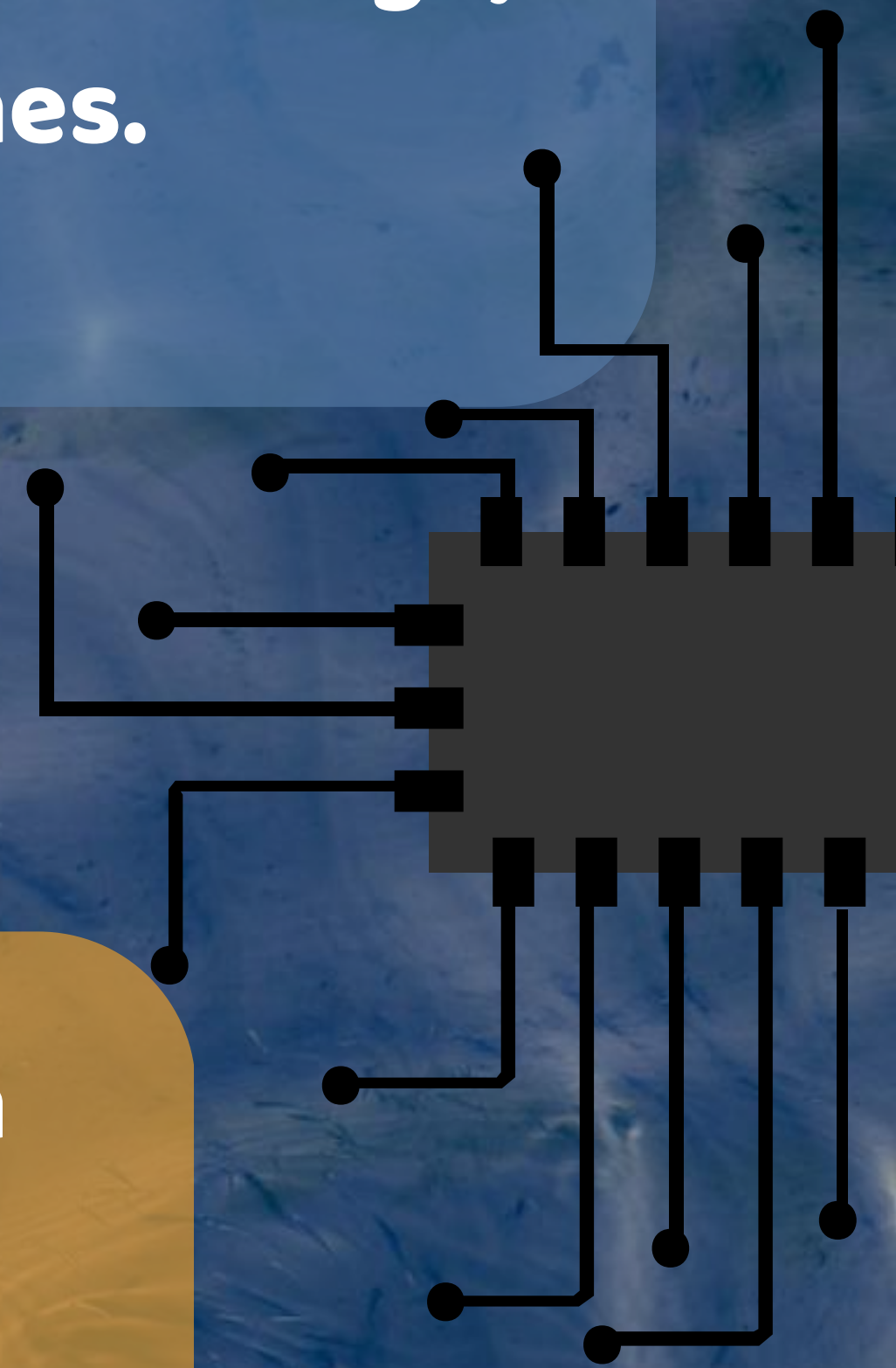
fecha: 28 de agosto del 2025

Introducción a Docker

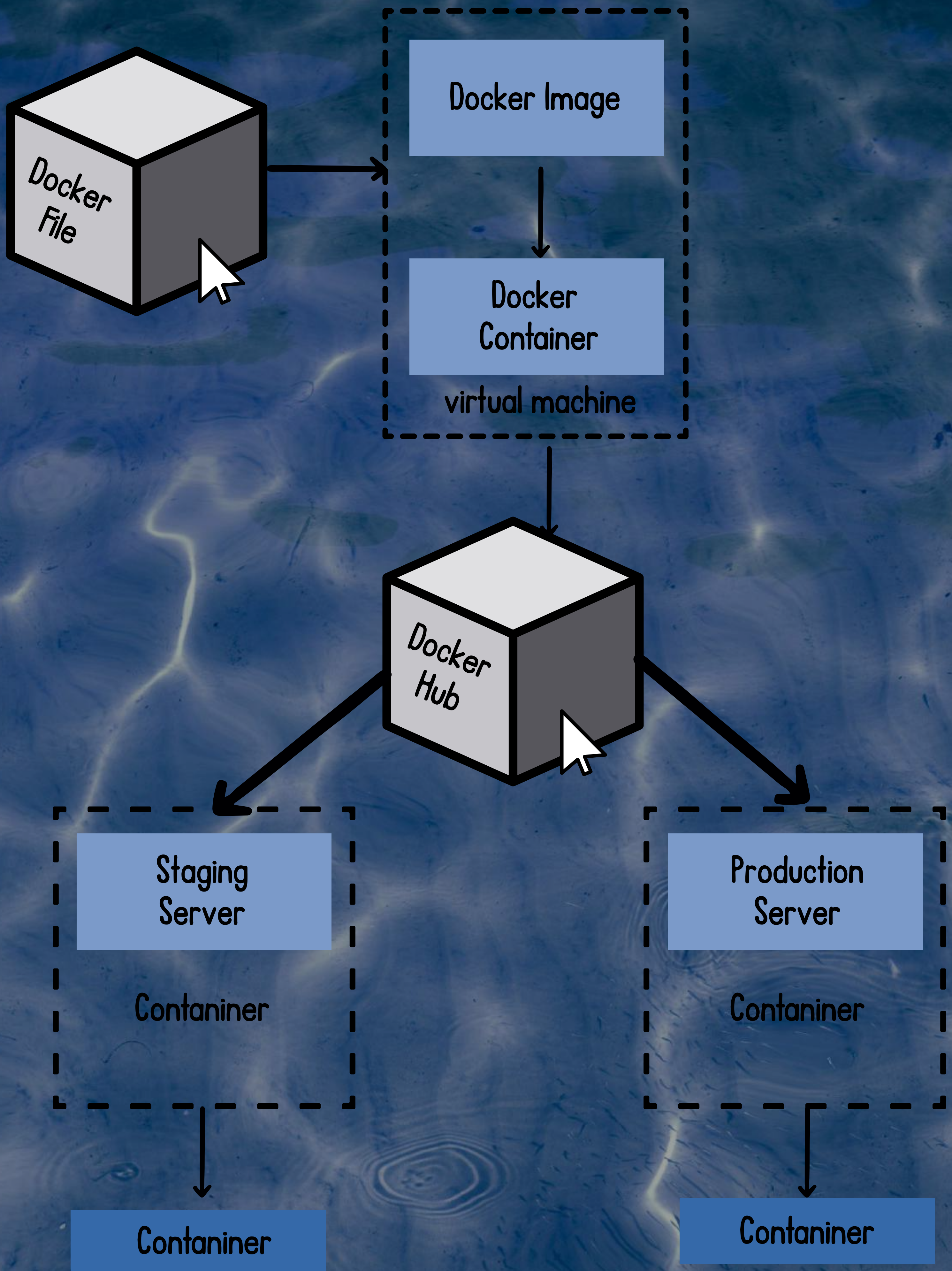
Docker es una plataforma de código abierto que permite desarrollar, enviar y ejecutar aplicaciones dentro de contenedores. Un contenedor es un entorno ligero y portable que incluye todo lo necesario para ejecutar una aplicación: código, bibliotecas, dependencias y configuraciones.

¿Por qué usar Docker

- **Aislamiento:** Las aplicaciones se ejecutan en entornos aislados.
- **Portabilidad:** Funciona en cualquier sistema con Docker instalado.
- **Eficiencia:** Los contenedores comparten el kernel del sistema operativo, por lo que son más ligeros que las máquinas virtuales.
- **CI/CD:** facilita la integración y despliegue continuo.






Conceptos Básicos









Comandos Básicos de Docker

Gestionar Imágenes

 Descargar	<code>docker pull <nombre_imagen></code>
 Listar	<code>docker images</code>
 Eliminar	<code>docker rmi <nombre_imagen></code>

Gestionar contenedores

 Ejecutar	<code>docker run -d -p <puerto_host>: <puerto_contenedor> --name <imagen></code>
 Listar (activos)	<code>docker ps</code>
 Listar (todos)	<code>docker ps -a</code>
 Detener	<code>docker stop <nombre_contenedor></code>
 Eliminar	<code>docker rm <nombre_contenedor></code>
 Ver logs	<code>docker logs <nombre_contenedor></code>

Construir y Publicar Imágenes



Construir

```
docker build -t <usuario>/<imagen>
```



Login

```
docker login
```



Subir

```
docker push <usuario>/<imagen>
```


¿Qué es un Dockerfile?

Es un archivo sin extensión que contiene una serie de instrucciones para construir una imagen de Docker.

Estructura básica



dockerfile

```
# Imagen base
FROM node:18

#Directorio de trabajo
WORKDIR / app

# Copiar archivos y instalar dependencias
COPY package*.json ./
RUN npm install

COPY . .

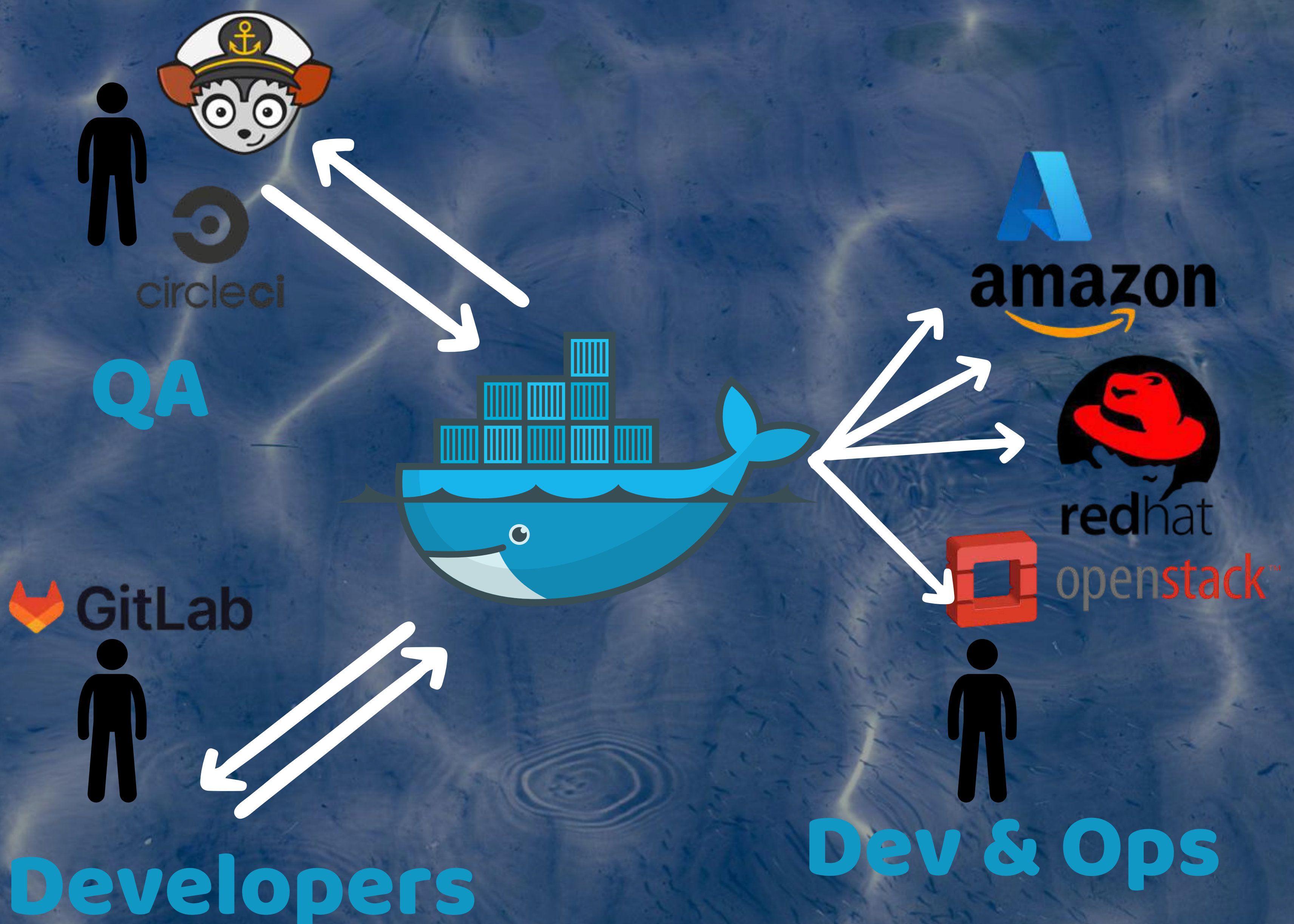
# Exponer puerto
EXPOSE 3000

# Comando para ejecutar
CMD ["npm", "start"]
```


¿Qué es Docker Hub?

Es el registro público de Docker donde se almacenan y comparten imágenes. Permite:

- Descargar imágenes preconstruidas (MySQL, MongoDB, Node.js, etc.).
- Subir tus propias imágenes.
- Colaborar con otros desarrolladores.



Docker Compose

Herramienta para definir y ejecutar aplicaciones multi-contenedor.

Ejemplo de docker-compose.yml:



dockerfile

```
version: '3'
services:
  web:
    image: tuusuario/app-cliente:1.0
    ports:
      - "80:80"
  api:
    image: tuusuario/app-servidor:1.0
    ports:
      - "3000:3000"
  db:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: ejemplo
```


Recomendaciones

- **Usa siempre tags específicos en lugar de latest.**
- **Limpia imágenes y contenedores que no uses.**
- **Usa variables de entorno para configuraciones sensibles.**
- **Documenta tus Dockerfiles con comentarios.**

Sección Práctica

Cliente - Servidor - Base de datos con DOCKER

1. Ejecutar una base de datos MySQL desde DockerHub

```
docker run --name mysql-db -e MYSQL_ROOT_PASSWORD=1234  
-e MYSQL_DATABASE=usuarios -d -p 3306:3306  
mysql:latest
```

2. Levantar la aplicación Servidor (API)

Ejemplo con Node.js (CRUD de usuario):

```
docker build -t miusuario/servidor-crud:1.0 ./servidor  
docker run -d -p 3000:3000 --name servidor --link  
mysql-db:mysql miusuario/servidor-crud:1.0
```

————→ La API se levanta en el puerto 3000 y está conectada al contenedor de MySQL.

Puedes probar con:

```
curl http://localhost:3000/users
```


3. Levantar la aplicación Cliente (React o Angular)

```
docker build -t miusuario/cliente-crud:1.0 ./cliente  
docker run -d -p 8080:80 --name cliente  
miusuario/cliente-crud:1.0
```

————→ El cliente se levanta en el puerto 8080 y consume la API en <http://localhost:300>

————→ Accede en el navegador: <http://localhost:8080>

4. Verificar contenedores en ejecución

```
docker ps
```

————→ Debes ver corriendo mysql-db, servidor y cliente

5. Ejemplo con Docker Compose

Archivo `docker-compose.yml`:



yml

```
version: '3'
services:
  db:
    image: mysql:latest
    environment:
      MYSQL_ROOT_PASSWORD: 1234
      MYSQL_DATABASE: usuarios
    ports:
      - "3306:3306"

  servidor:
    build: ./servidor
    ports:
      - "3000:3000"
    depends_on:
      - db

  cliente:
    build: ./cliente
    ports:
      - "8080:80"
    depends_on:
      - servidor
```

Levantar todo con:

```
docker compose up -d
```


6. Pruebas Finales

Cliente: Navegar a <https://localhost:8080>

Servidor: Probar un GET con curl <http://localhost:3000/users>

Base de datos: Conectarse con:

```
docker exec -it mysql-db mysql -u root -p
```

Enlaces Útiles

Documentación oficial de Docker: <https://docs.docker.com/>

Docker Hub: <https://hub.docker.com/>

Ejemplos de Dockerfiles: <https://dockerfile.github.io/>

FOTO GRUPAL

