

# DOCKERIZACIÓN Y DESPLIEGUE DE UN MICROSERVICIO EN AWS



PABLO DEL ÁLAMO

1/11



# INTRODUCCIÓN

- 
- Dockerizar un microservicio 🐳
  - Administrar imágenes Docker 📦
  - Desplegar en AWS usando varias opciones como ECS, EKS y Elastic Beanstalk 💻
  - Bonus: Comandos y mejores prácticas 🛠️



# ¿QUÉ ES DOCKER Y POR QUÉ USARLO?



- Docker es una plataforma que permite crear, enviar y ejecutar aplicaciones dentro de contenedores. La aplicación ahora corre dentro de un contenedor con x configuraciones, y por tanto, las configuraciones de tu máquina local o del servidor donde corre ya no importan.
- Beneficios de Docker:

1. Portabilidad: "Funciona en mi máquina" deja de ser un problema. Si despliegas el contenedor en tu local y funciona, garantizas que luego en QA o PROD también vaya a funcionar, al estar corriendo sobre un mismo contenedor
2. Aislamiento: Los contenedores evitan conflictos entre dependencias.
3. Escalabilidad: Facilita el despliegue en múltiples entornos



# PREPARANDO TU MICROSERVICIO PARA DOCKER



## Paso 1: Crear un archivo Dockerfile

- El Dockerfile define cómo Docker debe construir la imagen.

Ejemplo básico de Dockerfile en la siguiente diapositiva





```
# Imagen base de Node.js  
FROM node:14
```

```
# Crear directorio de la aplicación  
WORKDIR /app
```

```
# Copiar los archivos de tu proyecto  
COPY package*.json ./
```

```
# Instalar dependencias  
RUN npm install
```



```
# Copiar el resto del código  
COPY ..
```

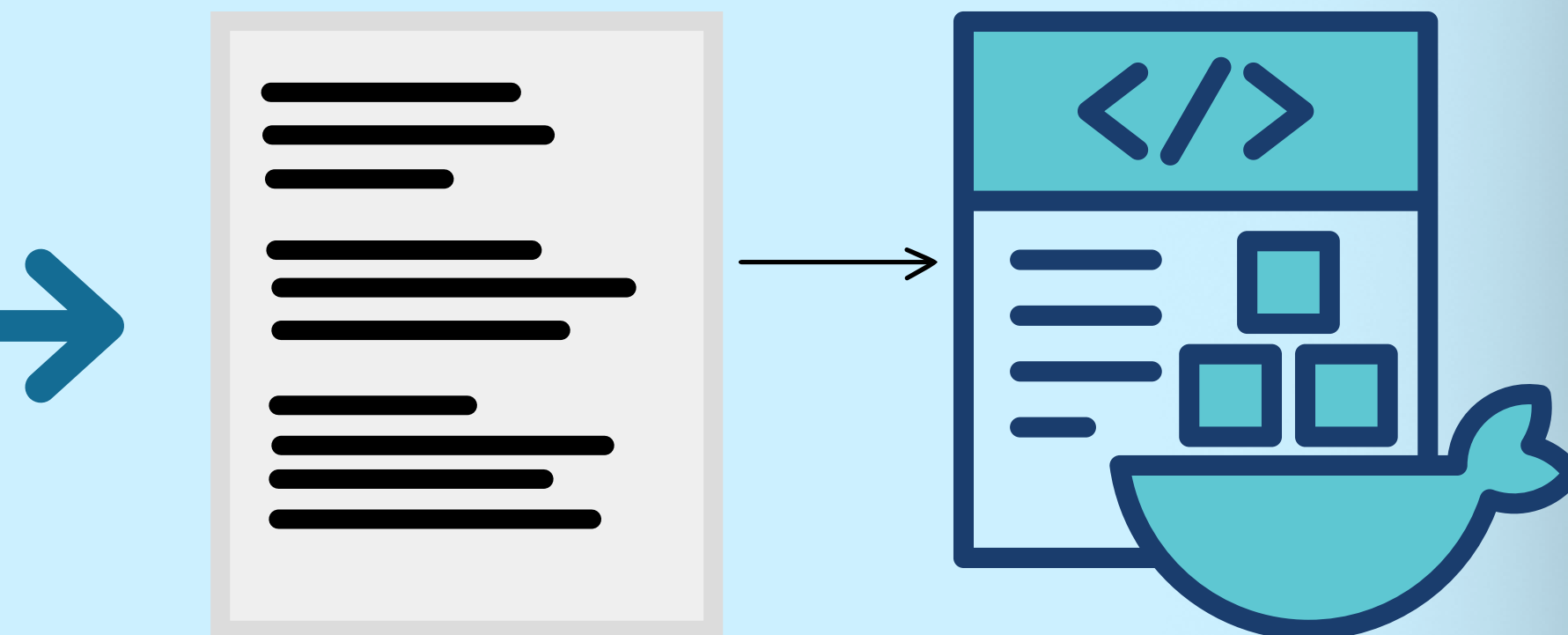
```
# Exponer el puerto  
EXPOSE 3000
```

```
# Comando para iniciar la aplicación  
CMD ["npm", "start"]
```





Paso 2: Construir la imagen  
`docker build -t mi-microservicio .`



# EJECUTAR Y PROBAR EL CONTENEDOR LOCALMENTE



Comando para ejecutar el contenedor:

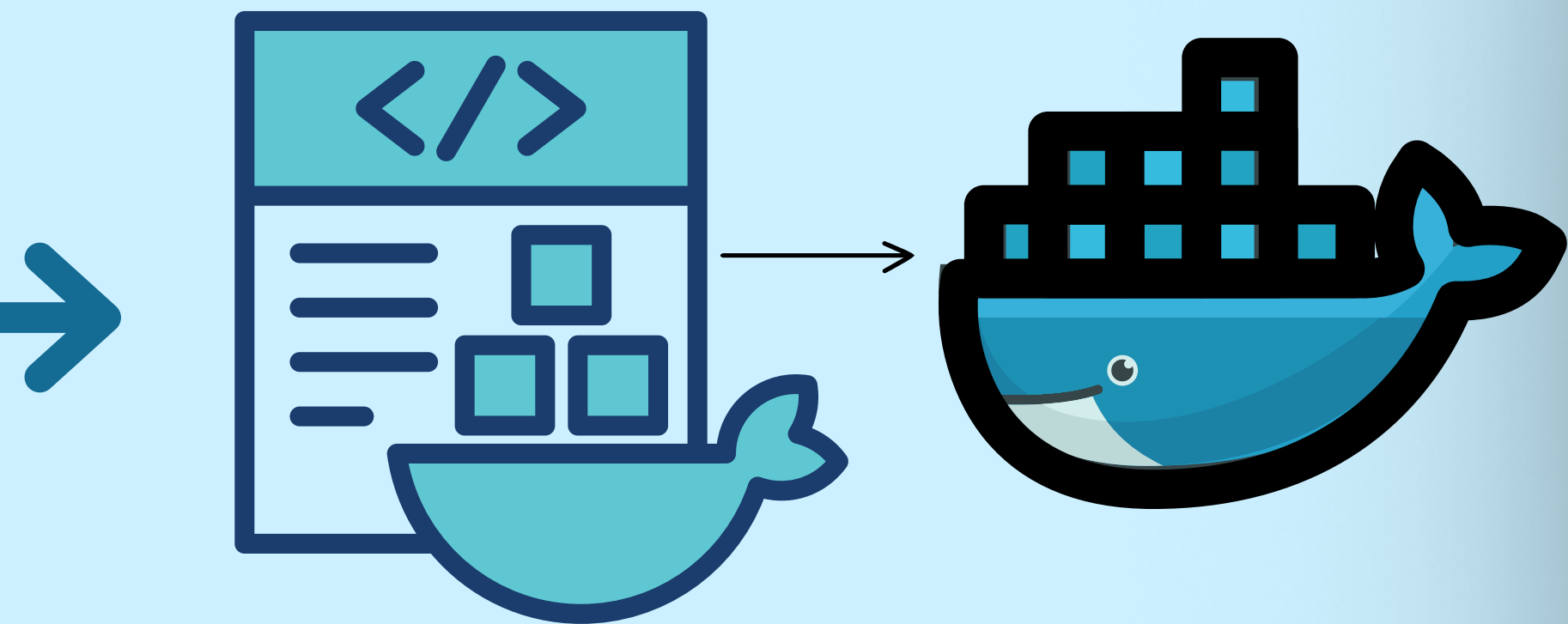
- `docker run -p 3000:3000 mi-microservicio`

Verificar que el contenedor está corriendo:

- `docker ps`

Tip: Asegúrate de que la aplicación corre sin problemas antes de pasar al siguiente paso.







# OPCIONES PARA ALMACENAR Y COMPARTIR IMÁGENES DOCKER



## 1. Docker Hub:

- El repositorio público más común para imágenes Docker.
- Comando para subir una imagen a Docker Hub:  
`docker tag mi-microservicio usuario/mi-microservicio, docker push usuario/mi-microservicio`

## 2. Amazon Elastic Container Registry (ECR):

- AWS ECR es un repositorio privado de imágenes Docker. Pasos para usar ECR:

1. Crear un repositorio en ECR.

2. Subir la imagen con el comando `docker push` hacia ECR.

# DESPLEGAR EL MICROSERVICIO EN AWS



A la hora de desplegar el microservicio en AWS tenemos diferentes opciones. Las 3 más usadas son:




- Elastic Beanstalk
- ECS (Elastic Container Service)
- EKS (Elastic Kubernetes Service)



# ELASTIC BEANSTALK



Elastic Beanstalk es un servicio de AWS que simplifica el despliegue y escalado de aplicaciones web y servicios desarrollados con tecnologías como Java, .NET, Node.js, Python, Ruby, Go y Docker.

- 
- ¿Cómo funciona?: Elastic Beanstalk se encarga de administrar automáticamente la infraestructura subyacente (servidores, balanceadores de carga, autoescalado, etc.) sin que tengas que preocuparte por la configuración manual.
  - Ventajas:
    - a. Administración simplificada: AWS gestiona la infraestructura, tú solo te concentras en el código.
    - b. Soporte para Docker: Ideal para aplicaciones contenedorizadas que necesitan un despliegue rápido.
    - c. Autoescalado: Escala automáticamente según la demanda.





# ELASTIC BEANSTALK

Pasos para desplegar con Elastic Beanstalk:



- Instalar Elastic Beanstalk CLI: `brew install awsebcli`
- Inicializar Elastic Beanstalk: `eb init`
- Crear un entorno de despliegue: `eb create mi-entorno`
- Desplegar la aplicación: `eb deploy`




# AMAZON ECS (ELASTIC CONTAINER SERVICE)

Amazon ECS es un servicio de orquestación de contenedores que te permite ejecutar y escalar aplicaciones en contenedores Docker en AWS.

- ➔ • ¿Cómo funciona?: ECS gestiona los contenedores Docker en clústeres de instancias EC2 o mediante AWS Fargate, una opción serverless que elimina la necesidad de administrar servidores.
- Dos modos de operación:
  - a. ECS con EC2: Ejecuta tus contenedores en instancias EC2. Ideal para quienes necesitan control completo sobre los servidores.
  - b. ECS con Fargate: No necesitas administrar servidores. Fargate se encarga de la infraestructura y solo pagas por los recursos que usas.




# AMAZON ECS (ELASTIC CONTAINER SERVICE)

- 
- Ventajas:
    - a. Alto control: Flexibilidad para elegir cómo ejecutar los contenedores.
    - b. Escalabilidad: Escala automáticamente según la carga.
    - c. Integración con ECR: Usa Amazon Elastic Container Registry (ECR) para almacenar imágenes Docker privadas.
  - Ejemplo: Definir y ejecutar una tarea ECS con Fargate:
  - `aws ecs run-task --cluster my-cluster --launch-type FARGATE --task-definition my-task`



# AMAZON ECS (ELASTIC CONTAINER SERVICE)

Pasos para desplegar con ECS:

- 
1. Crear un Cluster ECS.
  2. Definir una tarea ECS especificando la imagen Docker (de ECR o Docker Hub).
  3. Configurar el servicio ECS para ejecutar y escalar los contenedores.

Comando de ejemplo para subir a ECR:

```
aws ecr get-login-password --region region | docker  
login --username AWS --password-stdin  
<aws_account_id>.dkr.ecr.<region>.amazonaws.com  
docker tag mi-microservicio  
<aws_account_id>.dkr.ecr.  
<region>.amazonaws.com/mi-repo  
docker push <aws_account_id>.dkr.ecr.  
<region>.amazonaws.com/mi-repo
```





# EKS (ELASTIC KUBERNETES SERVICE)

Amazon EKS es un servicio administrado de Kubernetes que te permite ejecutar aplicaciones contenedorizadas en AWS usando Kubernetes, el popular sistema de orquestación de contenedores.

- ¿Cómo funciona?: EKS proporciona una configuración automática de Kubernetes para gestionar clústeres sin necesidad de manejar el plano de control.
- Ventajas:
  1. Escalabilidad avanzada: Kubernetes ofrece autoescalado y administración de grandes arquitecturas distribuidas.
  2. Portabilidad: Puedes mover aplicaciones entre distintos entornos Kubernetes (en la nube o en servidores locales).
  3. Ecosistema Kubernetes: Acceso a una vasta cantidad de herramientas de código abierto que funcionan con Kubernetes.





# EKS (ELASTIC KUBERNETES SERVICE)



- ¿Cuándo usarlo?: Ideal para empresas que ya están familiarizadas con Kubernetes o que necesitan orquestación avanzada para múltiples microservicios.



# EKS (ELASTIC KUBERNETES SERVICE)

Pasos para desplegar con EKS:

- Crear un cluster EKS con eksctl:
- `eksctl create cluster --name mi-cluster`
- Configurar tu archivo de despliegue de Kubernetes (deployment.yaml). Ejemplo básico de archivo deployment.yaml:



# EKS (ELASTIC KUBERNETES SERVICE)

apiVersion: apps/v1

kind: Deployment

metadata:

name: mi-app

spec:

replicas: 2

selector:

matchLabels:

app: mi-app

template:

metadata:

labels:

app: mi-app

spec:

containers:

- name: mi-app

image: <aws\_account\_id>.dkr.ecr.

<region>.amazonaws.com/mi-repo:latest

ports:

- containerPort: 80



# EKS (ELASTIC KUBERNETES SERVICE)



- Aplicar el despliegue: `kubectl apply -f deployment.yaml`



# COMPARACIÓN DE LAS DISTINTAS OPCIONES DE DESPLIEGUE



- Elastic Beanstalk: Sencillez y administración automática.
- ECS con Fargate: Sin servidores, adecuado para microservicios pequeños.
- ECS con EC2: Control total sobre la infraestructura.
- EKS (Kubernetes): Mejor para grandes arquitecturas distribuidas.



# CONCLUSIONES Y MEJORES PRÁCTICAS

- Usa Docker para garantizar que tu aplicación sea portátil.
- • Elige la opción de despliegue en AWS que mejor se ajuste a tus necesidades.
- Automatiza el proceso de despliegue con herramientas como AWS CodePipeline para mejorar la eficiencia.



# ¿TE HA RESULTADO ÚTIL?



➔ ¡Has completado el proceso de dockerización y despliegue en AWS! 🚀

- Comparte esta guía con tu equipo o amigos desarrolladores.
- Guárdala para tener siempre a mano los pasos.
- ¡Dale un like o comenta si tienes preguntas!

