

UNIVERSIDAD INTERNACIONAL DE LA RIOJA (UNIR)

La Universidad en Internet

Contenedores

(MEXDEVOPS)

**Maestría en Desarrollo y Operaciones de Software
2025**

Actividad 3 grupal: Construcción de aplicativo en
contenedores y despliegue en un clúster de
Kubernetes

Equipo 2_H

Índice

Generar las imágenes de los aplicativos con Dockerfile	3
Estructura basada en contenedores: separación de responsabilidades.....	3
Dockerfile App	3
Dockerfile DB	3
Configuración con docker-compose.yml para hacer pruebas.....	3
Pruebas locales	4
Publicación en Docker Hub.....	5
Kubernetes	5
Resultado:	8
Limpiar espacio local	8
EKS (Amazon Elastic Kubernetes Service)	8
Permisos AWS CLI	9
Creación del clúster.....	9
Comprobación de creación y contexto del clúster.....	10
Aplicación de los manifiestos dentro de nuestro clúster.....	11
Resultados finales	12
Conclusión	13
Referencias	13

Generar las imágenes de los aplicativos con Dockerfile

Estructura basada en contenedores: separación de responsabilidades

Como ya se ha mencionado, para esta actividad se ha diseñado una aplicación compuesta por dos contenedores:

- Contendor App: expone una API Rest construida con Node.js y Express. Consulta una base de datos para recuperar diferente información entre ella productos.
- Contenedor DB: instancia PostgreSQL con datos precargados mediante un script SQL (init.sql)

Esta arquitectura refleja el principio de separación de responsabilidades, facilita el despliegue, la escalabilidad independiente de los servicios y una migración sencilla a un clúster de Kubernetes.

Dockerfile App

Para este Dockerfile se usa una imagen base minimalista la cual reduce el tamaño final, un usuario sin privilegios para mejorar la seguridad para evitar la ejecución como modo root, separación de dependencias, exposición explícita del puerto 3000, uso de npm ci haciendo una instalación determinista basada en package-lock.json ideal para producción.

Dockerfile DB

Este Dockerfile extiende la imagen oficial de PostgreSQL y añade un script SQL para inicializar una tabla de productos con datos de prueba. Esto permite a que la API tenga datos desde el inicio y sea más realista su integración en Kubernetes.

Configuración con docker-compose.yml para hacer pruebas

Se crea un archivo compose para hacer la prueba de que todo este funcionando correctamente, las principales decisiones para esta configuración son:

Separación en servicios: se definen servicios independientes (app y db).

Red compartida personalizada (actividad3-net): permite que los contenedores se comuniquen usando el nombre del servicio como host (DB_HOST = db)

Variables de retorno explícitas: se utilizan para definir credenciales, puertos y nombres de base de datos, facilitando su futura gestión con Secrets en Kubernetes.

Montaje de un volumen persistente (db-data): asegura que los datos de PostgreSQL no se pierdan al reiniciar el contenedor, lo que es un paso intermedio antes de definir PersistentVolumeClaims en Kubernetes.

Mapeo de puertos (3000:3000): permite exponer el contenedor de la aplicación localmente a través del puerto que nosotros elijamos, facilitando pruebas manuales.

Pruebas locales

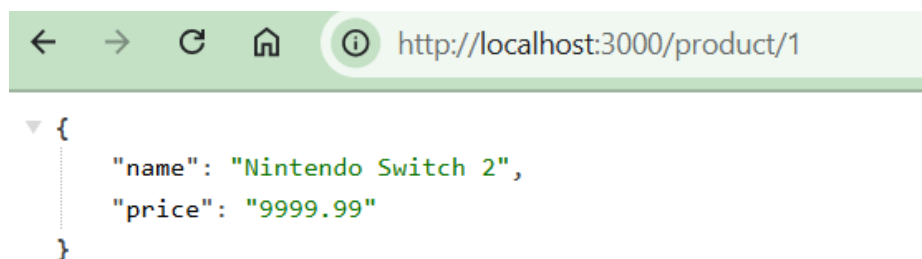
Hacemos un docker compose up --build para poder iniciar nuestro proyecto

```
PS D:\UNIR\Maestria\unir\2 semestres\Contenedores\actividad3> docker compose up --build
time="2025-06-19T12:52:25-06:00" level=warning msg="D:\UNIR\Maestria\unir\2 semestres\Contenedores\actividad3\docker-
compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Building 2.8s (22/22) FINISHED                                docker:desktop-linux
=> [db internal] load build definition from Dockerfile            0.0s
=> => transferring dockerfile: 205B                               0.0s
=> [db internal] load metadata for docker.io/library/postgres:16-alpine 1.1s
=> [db auth] library/postgres:pull token for registry-1.docker.io 0.0s
=> [db internal] load .dockerignore                               0.0s
=> => transferring context: 2B                                     0.0s
=> [db internal] load build context                               0.0s
=> => transferring context: 30B                                    0.0s
=> [db 1/2] FROM docker.io/library/postgres:16-alpine@sha256:ef2235fd13b6cb29728a98ee17862ff5c9b7d20515a9b34804da4a 0.0s
=> CACHED [db 2/2] COPY init.sql /docker-entrypoint-initdb.d/    0.0s
```

Podemos comprobar en la consola que todo esté en orden y también en nuestra interfaz de Docker Desktop:

<input type="checkbox"/>	▼	●	actividad3	-	-	-			
<input type="checkbox"/>		●	db	e756be870976	actividad3-				
<input type="checkbox"/>		●	app	7fce1ac49f7b	actividad3- 3000:3000				

Por último de manera local podemos comprobar que la aplicación web funciona correctamente y también puede conectarse al contenedor de la base de datos:



Para detener nuestros contenedores podemos hacer uso del comando docker compose down.

Publicación en Docker Hub

Una vez validada las imágenes localmente se procede a la publicación en Docker Hub. Esto es necesario para garantizar que los manifiestos de Kubernetes puedan acceder a imágenes públicas sin necesidad de reconstruirlas en el entorno distinto.

Los pasos realizados fueron:

1. Se etiquetan las imágenes locales con el nombre del repositorio en Docker Hub utilizando:
`docker tag actividad3-app:latest lemendezc/actividad3-app:1.0.0`
`docker tag actividad3-db:latest lemendezc/actividad3-db:1.0.0`
2. Nos autenticamos con `docker login`.
3. Se realiza el envío (push) de ambas imágenes (actividad3-app y actividad3-db) con una versión explícita 1.0.0.

Name	Last Pushed 	Contains	Visibility	Scout
lemendezc/actividad3-db	22 minutes ago		Public	Inactive
lemendezc/actividad3-app	22 minutes ago		Public	Inactive

Kubernetes

Para trabajar con Kubernetes antes de hacerlo directamente en una nube como AWS haremos y configuraremos todo para que funcione localmente.

- El primer archivo es 01-namespace.yaml, el cual define un recurso de tipo Namespace dentro del clúster de Kubernetes.
- El archivo 02-secret.yaml define un recurso de tipo Secret, cuya finalidad es almacenar de forma segura las credenciales de acceso a la base de datos PostgreSQL, gestiona la contraseña de la base de datos de manera separada y segura.
- El archivo 03-pvc-db.yaml define un recurso de tipo PersistentVolumeClaim (PVC), que solicita a Kubernetes un volumen persistente de 1 GiB para almacenar los datos de la base de datos PostgreSQL
- 04-deployment-db.yaml este archivo define un Deployment de Kubernetes para desplegar la base de datos PostgreSQL como un pod controlado y administrado automáticamente.

La configuración del contenedor incluye:

- `envFrom.secretRef`: importa las variables de entorno necesarias (`POSTGRES_DB`, `POSTGRES_USER`, `POSTGRES_PASSWORD`) desde el Secret previamente creado (`db-secret`), evitando así escribir valores sensibles directamente en el manifiesto.
- `ports.containerPort`: 5432: expone el puerto estándar de PostgreSQL dentro del contenedor.
- `volumeMounts` + `volumes`: monta el PVC (`db-pvc`) en la ruta `/var/lib/postgresql/data`, que es el directorio donde PostgreSQL guarda sus datos. Esto garantiza persistencia aunque el contenedor sea destruido y recreado.
- `05-deployment-app.yaml` este manifiesto define el Deployment de la aplicación web, es decir, el componente que expone la API REST desarrollada en Node.js.
- `06-service-db.yaml` este archivo define un recurso de tipo Service que permite exponer el pod de PostgreSQL a otros recursos dentro del clúster de Kubernetes. El servicio se llama `postgres` y está asociado al namespace `actividad3`.
- `7-service-app.yml` este manifiesto define un recurso de tipo Service para exponer la aplicación web al exterior del clúster. A diferencia del servicio de base de datos, este Service está pensado para ser accedido por usuarios desde fuera de Kubernetes. permite acceder desde fuera del clúster
- `08-pv-db.yaml` necesario para configurar el PersistentVolume especificando el tamaño, id del volumen y la región.

Localmente: Aplicar los manifiestos en orden

Namespace:

```
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> kubectl apply -f 01-namespace.yaml
namespace/actividad3 created
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> kubectl get namespaces
>>
NAME                STATUS    AGE
actividad3          Active    50s
default              Active    195d
kube-node-lease      Active    195d
kube-public          Active    195d
kube-system          Active    195d
```

Secret:

```
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> kubectl apply -f 02-secret.yaml
secret/db-secret configured
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> kubectl get secret db-secret -n actividad3
```

NAME	TYPE	DATA	AGE
db-secret	Opaque	3	42s

PersistentVolumeClaim:

```
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> kubectl apply -f 03-pvc-db.yaml
persistentvolumeclaim/db-pvc created
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> kubectl get pvc -n actividad3
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	VOLUMEATTRIBUTESCLASS	AGE
db-pvc	Bound	pvc-7e8fc28d-97a7-4725-9da3-0fa8e8139afd	1Gi	RWO	hostpath	<unset>	6s

Deployment DB:

```
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> kubectl apply -f 04-deployment-db.yaml
deployment.apps/postgres created
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> kubectl get deployments -n actividad3
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
postgres	1/1	1	1	6s

Service DB:

```
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> kubectl apply -f 06-service-db.yaml
service/postgres created
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> kubectl get svc -n actividad3
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
postgres	ClusterIP	10.98.128.187	<none>	5432/TCP	20s

Deployment App:

```
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> kubectl apply -f 05-deployment-app.yaml
deployment.apps/app created
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> kubectl get deployments -n actividad3
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
app	2/2	2	2	25s
postgres	1/1	1	1	3m4s

Service App:

```
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> kubectl apply -f 07-service-app.yaml
service/actividad3-app created
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> kubectl get svc -n actividad3
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
actividad3-app	NodePort	10.101.251.245	<none>	80:30036/TCP	12s
postgres	ClusterIP	10.98.128.187	<none>	5432/TCP	2m30s

Resultado:



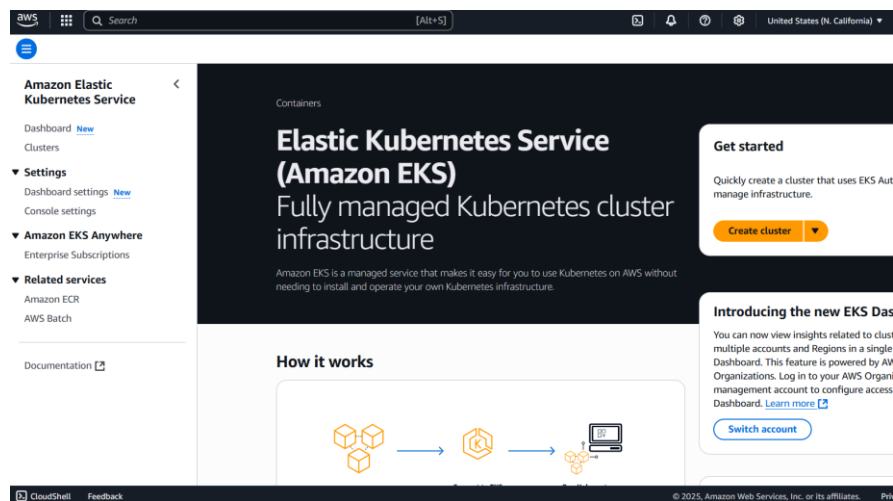
Limpiar espacio local

Ya que todo fue creado dentro del namespace actividad3 es sencillo eliminar todo lo creado con el siguiente comando:

kubectl delete namespace actividad3

```
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> kubectl delete namespace actividad3
namespace "actividad3" deleted
```

EKS (Amazon Elastic Kubernetes Service)



Instalación de eksctl para poder crear el cluster de kubernetes desde la consola de nuestra máquina.

choco install eksctl -y


```

PS C:\Users\lemen> choco install eksctl -y
Chocolatey v2.4.1
Installing the following packages:
eksctl
By installing, you accept licenses for the packages.
Downloading package from source 'https://community.chocolatey.org/api/v2/'

eksctl v0.210.0 [Approved]
eksctl package files install completed. Performing other installation steps.
eksctl is going to be installed in 'C:\ProgramData\chocolatey\lib\eksctl\tools'
Downloading eksctl 64 bit
  from 'https://github.com/eksctl-io/eksctl/releases/download/v0.210.0/eksctl_Windows_amd64.zip'
Progress: 100% - Completed download of C:\Users\lemen\AppData\Local\Temp\chocolatey\eksctl\0.210.0\eksctl_Windows_amd64.zip (34.14 MB).
Download of eksctl_Windows_amd64.zip (34.14 MB) completed.
Hashes match.
Extracting C:\Users\lemen\AppData\Local\Temp\chocolatey\eksctl\0.210.0\eksctl_Windows_amd64.zip to C:\ProgramData\chocolatey\lib\eksctl\tools...
C:\ProgramData\chocolatey\lib\eksctl\tools
ShimGen has successfully created a shim for eksctl.exe
The install of eksctl was successful.
  Deployed to 'C:\ProgramData\chocolatey\lib\eksctl\tools'

Chocolatey installed 1/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).

```

Permisos AWS CLI


Hay que tener en cuenta que es posible que el usuario que tenemos logeado en nuestra consola aws cli en nuestra computadora puede no tener los permisos necesarios para ello hay que asignar los siguientes permisos:

User details

User name
packer-user

Permissions summary (7)

< 1 >

Name 	Type	Used as
AmazonEKSClusterPolicy	AWS managed	Permissions policy
AmazonEKSServicePolicy	AWS managed	Permissions policy
AmazonECS_FullAccess	AWS managed	Permissions policy
IAMFullAccess	AWS managed	Permissions policy
AmazonEKSWorkerNodePolicy	AWS managed	Permissions policy
AmazonEKS_CNI_Policy	AWS managed	Permissions policy
AmazonEC2ContainerRegistryReadOnly	AWS managed	Permissions policy

En nuestro caso fue necesario agregar un política un poco más laxa para el laboratorio y fue creado con json.

Creación del clúster

Ahora podemos crear el clúster con un comando en PowerShell:

```
eksctl create cluster --name actividad3-cluster --region us-west-1 --version 1.29 --nodegroup-name actividad3-nodes --node-type t3.medium --nodes 2 --nodes-min 1 --nodes-max 3 --managed
```

Esto creará:

- una VPC con subredes y seguridad,

- un clúster EKS con control plane gestionado,
- un grupo de nodos EC2 administrado (2 instancias t3.medium),
- y actualizará el archivo ~/.kube/config para que kubectl ya pueda usarlo directamente.

Clusters (1) [Info](#)

Filter clusters

	Cluster name	Status	Kubernetes version	Support period	Upgrade policy	Created	Provider
<input type="radio"/>	actividad3-cluster	Active	1.29 Upgrade now	Extended support until March 22, 2026	Extended	7 minutes ago	EKS

```

2025-06-19 14:43:40 [i] creating addon: vpc-cni
2025-06-19 14:43:40 [i] successfully created addon: vpc-cni
2025-06-19 14:43:41 [i] creating addon: kube-proxy
2025-06-19 14:43:41 [i] successfully created addon: kube-proxy
2025-06-19 14:45:43 [i] building managed nodegroup stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes"
2025-06-19 14:45:44 [i] deploying stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes"
2025-06-19 14:45:44 [i] waiting for CloudFormation stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes"
2025-06-19 14:46:14 [i] waiting for CloudFormation stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes"
2025-06-19 14:46:58 [i] waiting for CloudFormation stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes"
2025-06-19 14:48:37 [i] waiting for CloudFormation stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes"
2025-06-19 14:49:39 [i] waiting for CloudFormation stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes"
2025-06-19 14:49:39 [i] waiting for the control plane to become ready
2025-06-19 14:49:41 [i] saved kubeconfig as "C:\\Users\\lemon\\.kube\\config"
2025-06-19 14:49:41 [i] no tasks
2025-06-19 14:49:41 [i] all EKS cluster resources for "actividad3-cluster" have been created
2025-06-19 14:49:41 [i] nodegroup "actividad3-nodes" has 2 node(s)
2025-06-19 14:49:41 [i] node "ip-192-168-15-244.us-west-1.compute.internal" is ready
2025-06-19 14:49:41 [i] node "ip-192-168-60-4.us-west-1.compute.internal" is ready
2025-06-19 14:49:41 [i] waiting for at least 1 node(s) to become ready in "actividad3-nodes"
2025-06-19 14:49:41 [i] nodegroup "actividad3-nodes" has 2 node(s)
2025-06-19 14:49:41 [i] node "ip-192-168-15-244.us-west-1.compute.internal" is ready
2025-06-19 14:49:41 [i] node "ip-192-168-60-4.us-west-1.compute.internal" is ready
2025-06-19 14:49:41 [i] created 1 managed nodegroup(s) in cluster "actividad3-cluster"
2025-06-19 14:49:44 [i] kubectl command should work with "C:\\Users\\lemon\\.kube\\config", try 'kubectl get nodes'
2025-06-19 14:49:44 [i] EKS cluster "actividad3-cluster" in "us-west-1" region is ready

```

Comprobación de creación y contexto del clúster

Primero comprobamos los nodos:

```

PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3> kubectl get nodes
NAME
ip-192-168-15-244.us-west-1.compute.internal   Ready    <none>    3m36s    v1.29.15-eks-473151a
ip-192-168-60-4.us-west-1.compute.internal   Ready    <none>    3m40s    v1.29.15-eks-473151a
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3>

```

Node groups (1) [Info](#)

Node groups implement basic compute scaling through EC2 Auto Scaling groups.

Edit Delete [Add node group](#)

	Group name	Desired size	AMI release version	Launch template	Status
<input type="radio"/>	actividad3-nodes	2	1.29.15-20250610	eksctl-actividad3-cluster-nodegroup-actividad3-nodes (1)	Active

Node group configuration [Info](#)

Kubernetes version 1.29	AMI type Info Amazon Linux 2 (x86_64)	Launch template eksctl-actividad3-cluster-nodgroup-actividad3-nodes	Status Active
AMI release version Info 1.29.15-20250610	Instance types t3.medium	Launch template version 1	Disk size Specified in launch template

[Details](#)
[Nodes](#)
[Health issues](#) 0
[Kubernetes labels](#)
[Update config](#)
[Kubernetes taints](#)
[Update history](#)
[Tags](#)

Details

Node group ARN
[arn:aws:eks:us-west-1:396608782191:nodegroup/actividad3-cluster/actividad3-nodes/6ccbc4f8-8572-ba98-7b10-3fab1bfe3b06](#)

Created
7 minutes ago

Autoscaling group name
[eks-actividad3-nodes-6ccbc4f8-8572-ba98-7b10-3fab1bfe3b06](#)

Node IAM role ARN
[arn:aws:iam::396608782191:role/eksctl-actividad3-cluster-nodgroup-NodeInstanceRole-0sfaFn88vqvM](#)
[View in IAM](#)

Capacity type
On-Demand

Desired size
2 nodes

Minimum size
1 node

Maximum size
3 nodes

Subnets
[subnet-01704d6857032c6e2](#)
[subnet-0a541ccedfb33a125](#)

Configure remote access to nodes
off

Nos tenemos que dar cuenta que estemos en el contexto adecuado:

kubectl config get-contexts

```
PS D:\UNIR\Maestria\unir\2 semestres\Contenedores\actividad3\k8s> kubectl config get-contexts
CURRENT  NAME                                CLUSTER                                AUTHINFO
*         packer-user@actividad3-cluster.us-west-1.eksctl.io  actividad3-cluster.us-west-1.eksctl.io  packer-user@actividad3-cluster.us-west-1.eksctl.io
```

Aplicación de los manifiestos dentro de nuestro clúster

Dentro de nuestra carpeta k8s corremos el siguiente comando:

kubectl apply -f .\

Nota: Es importante hacer cambio al archivo 07-service-app.yaml ya que haremos uso de un LoadBalancer.

Resultado:

```
PS D:\UNIR\Maestria\unir\2 semestres\Contenedores\actividad3\k8s> kubectl apply -f 01-namespace.yaml
>> kubectl apply -f 02-secret.yaml
>> kubectl apply -f 03-pvc-db.yaml
>> kubectl apply -f 04-deployment-db.yaml
>> kubectl apply -f 05-deployment-app.yaml
>> kubectl apply -f 06-service-db.yaml
>> kubectl apply -f 07-service-app.yaml
>>
namespace/actividad3 created
secret/db-secret created
persistentvolumeclaim/db-pvc created
persistentvolume/db-pv created
deployment.apps/postgres created
service/postgres created
deployment.apps/app created
service/actividad3-app created
PS D:\UNIR\Maestria\unir\2 semestres\Contenedores\actividad3\k8s>
```

Crear un volumen para después ser montado, al igual que agregar un addon dentro de nuestro clúster the ebs csi.

Successfully created volume vol-040b31c191757f0e2.

Volumes (1/3) Info

Last updated less than a minute ago

Actions Create volume

Choose filter set Search

	Name	Volume ID	Type	Size	IOPS	Throughput	Snapshot ID	Created	Availability Z
<input type="checkbox"/>	actividad3-cluster-actividad3...	vol-0061e53fc1df49db0	gp3	80 GiB	3000	125	snap-09de109...	2025/06/19 14:47 GMT-6	us-west-1a
<input checked="" type="checkbox"/>	vol-actividad3-luis	vol-040b31c191757f0e2	gp2	1 GiB	100	-	-	2025/06/19 15:07 GMT-6	us-west-1a
<input type="checkbox"/>	actividad3-cluster-actividad3...	vol-0e5be0dca82de80dd	gp3	80 GiB	3000	125	snap-09de109...	2025/06/19 14:47 GMT-6	us-west-1b

```
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> eksctl create addon `
>> --name "aws-ebs-csi-driver" `
>> --cluster "actividad3-cluster" `
>> --region "us-west-1" `
>> --service-account-role-arn "arn:aws:iam::396608782191:role/AmazonEKS_EBS_CSI_DriverRole" `
>> --force
>>
2025-06-19 15:36:33 [i] Kubernetes version "1.29" in use by cluster "actividad3-cluster"
2025-06-19 15:36:34 [!] IRSA config is set for "aws-ebs-csi-driver" addon, but since OIDC is disabled on the cluster, eksctl cannot
configure the requested permissions; the recommended way to provide IAM permissions for "aws-ebs-csi-driver" addon is via pod identity
y associations; after addon creation is completed, add all recommended policies to the config file, under `addon.PodIdentityAssociat
ions`, and run `eksctl update addon`
2025-06-19 15:36:34 [i] creating addon: aws-ebs-csi-driver
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s>
```

Resultados finales

Obtenemos la dirección con el siguiente comando:

kubectl get svc actividad3-app -n actividad3

```
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> kubectl get svc actividad3-app -n actividad3
NAME          TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
actividad3-app LoadBalancer  10.100.84.145    a7213907058ad4e4a7f0f1638f2701-612352830.us-west-1.elb.amazonaws.com  80:31506/TCP    64s
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s>
```

<http://abff2f8c78fac4fe4874d16f34649353-2026088694.us-west-1.elb.amazonaws.com/health>

← → ↻ 🏠 ⚠ Not secure http://abff2f8c78fac4fe4874d16f34649353-2026088694.us-west-1.elb.amazonaws.com/health

OK

← → ↻ 🏠 ⚠ Not secure http://abff2f8c78fac4fe4874d16f34649353-2026088694.us-west-1.elb.amazonaws.com/product/1

```
{
  "name": "Nintendo Switch 2",
  "price": "9999.99"
}
```

Eliminar el contenido y después el clúster completamente:

```
PS D:\UNIR\Maestria\unir\2_semestre\Contenedores\actividad3\k8s> kubectl delete -f .
namespace "actividad3" deleted
secret "db-secret" deleted
persistentvolumeclaim "db-pvc" deleted
deployment.apps "postgres" deleted
deployment.apps "app" deleted
service "postgres" deleted
service "actividad3-app" deleted
persistentvolume "db-pv" deleted
```

eksctl delete cluster --name actividad3-cluster --region us-west-1

```
PS D:\UNIR\Maestria\unir\2 semestres\Contenedores\actividad3\k8s> eksctl delete cluster --name actividad3-cluster --region us-west-1
>>
2025-06-19 16:00:38 [i] deleting EKS cluster "actividad3-cluster"
2025-06-19 16:00:39 [i] will drain 0 unmanaged nodegroup(s) in cluster "actividad3-cluster"
2025-06-19 16:00:39 [i] starting parallel draining, max in-flight of 1
2025-06-19 16:00:40 [i] deleted 0 Fargate profile(s)
2025-06-19 16:00:41 [✓] kubeconfig has been updated
2025-06-19 16:00:41 [i] cleaning up AWS load balancers created by Kubernetes objects of Kind Service or Ingress
2025-06-19 16:00:42 [i]
2025-06-19 16:00:43 [i] deleting parallel draining, max in-flight of 1
2025-06-19 16:00:43 [i] deleted 0 Fargate profile(s)
2025-06-19 16:00:41 [✓] kubeconfig has been updated
2025-06-19 16:00:41 [i] cleaning up AWS load balancers created by Kubernetes objects of Kind Service or Ingress
2025-06-19 16:00:42 [i]
2 sequential tasks: { delete nodegroup "actividad3-nodes", delete cluster control plane "actividad3-cluster" [async]
}
2025-06-19 16:00:43 [i] will delete stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes"
2025-06-19 16:00:43 [i] waiting for stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes" to get deleted
2025-06-19 16:00:43 [i] waiting for CloudFormation stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes"
2025-06-19 16:01:13 [i] waiting for CloudFormation stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes"
2025-06-19 16:01:49 [i] waiting for CloudFormation stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes"
2025-06-19 16:03:13 [i] waiting for CloudFormation stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes"
2025-06-19 16:04:28 [i] waiting for CloudFormation stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes"
2025-06-19 16:06:13 [i] waiting for CloudFormation stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes"
2025-06-19 16:07:46 [i] waiting for CloudFormation stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes"
2025-06-19 16:09:18 [i] waiting for CloudFormation stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes"
2025-06-19 16:10:11 [i] waiting for CloudFormation stack "eksctl-actividad3-cluster-nodegroup-actividad3-nodes"
2025-06-19 16:10:11 [i] will delete stack "eksctl-actividad3-cluster-cluster"
2025-06-19 16:10:12 [✓] all cluster resources were deleted
PS D:\UNIR\Maestria\unir\2 semestres\Contenedores\actividad3\k8s>
```

Conclusión

La implementación de una aplicación web con base de datos en un clúster de Kubernetes permitió comprender a fondo la gestión de volúmenes persistentes, servicios, despliegues y namespaces. Además, se fortalecieron habilidades clave para el manejo de errores, la depuración de contenedores y la eliminación segura de recursos en entornos productivos y de desarrollo.

Referencias

- Hightower, K., Burns, B., & Beda, J. (2017). Kubernetes: Up and Running: Dive into the Future of Infrastructure. O'Reilly Media.
- The Kubernetes Authors. (2024). Kubernetes Documentation. Retrieved from <https://kubernetes.io/docs/>
- Amazon Web Services. (2024). Amazon EKS User Guide. Retrieved from <https://docs.aws.amazon.com/eks/>