

EKS: Kubernetes en AWS

Amazon Elastic Kubernetes Service — Kubernetes sin tener que mantener Kubernetes

EL PROBLEMA DE INSTALAR KUBERNETES VOS MISMA

Kubernetes es poderoso, pero el **Control Plane** (el cerebro) tiene componentes críticos que hay que instalar, configurar, asegurar y mantener:

- **API Server** — si se cae, nadie puede desplegar ni ver el estado del cluster
- **etcd** — la base de datos del cluster. Si se corrompe y no hay backup, perdés todo
- **Certificados TLS** — hay que generarlos, rotarlos, y que no se venzan
- **Actualizaciones** — K8s saca versiones nuevas cada 4 meses, y hay que migrar sin downtime

Mantener todo eso en alta disponibilidad es un trabajo de equipo dedicado. Muchas empresas no tienen ese equipo — y no deberían necesitarlo.

¿QUÉ ES EKS?

EN UNA FRASE

Amazon EKS (Elastic Kubernetes Service) es Kubernetes como servicio: AWS gestiona el Control Plane por vos. Vos solo te ocupás de tus aplicaciones y los archivos YAML que las describen.

Es **Kubernetes estándar** — no una versión modificada de AWS. Los mismos archivos YAML que funcionan en tu laptop con minikube, funcionan en EKS sin cambiar nada.

VOS	Tus aplicaciones en Pods (los contenedores) Los archivos YAML que describen tus Deployments y Services Los worker nodes donde corren los Pods (o Fargate)
AWS GESTIONA	API Server — siempre disponible, multi-AZ, redundante etcd — backups automáticos cada 5 min, encriptado en reposo Actualizaciones de versión de Kubernetes Certificados TLS — rotación automática Parches de seguridad del Control Plane

CÓMO SE CREA UN CLUSTER EKS

`eksctl` es la herramienta oficial de línea de comandos para crear y administrar clusters EKS. Es a EKS lo que `docker` es a Docker.

```
# Crear un cluster con 2 worker nodes
eksctl create cluster \
  --name mi-cluster \
  --region us-east-1 \
  --nodes 2 \
  --node-type t3.medium
```

Eso es todo. En ~15 minutos, AWS crea:

- Un **Control Plane** gestionado (API Server + etcd + Scheduler + Controller)
- 2 instancias **EC2 t3.medium** como worker nodes
- Una **VPC** con subnets, security groups, y roles IAM configurados
- Tu `kubectl` ya configurado para hablar con el cluster

DESPUÉS DE CREAR EL CLUSTER

Podés usar `kubectl` normalmente: `kubectl get nodes` para ver los worker nodes, `kubectl apply -f` para desplegar. Es Kubernetes estándar — todo lo que aprendiste aplica igual.

COMANDOS ESENCIALES DE EKSCTL

CLUSTER

<code>eksctl create cluster --name X</code>	Crea un cluster nuevo con configuración por defecto
<code>eksctl get cluster</code>	Lista los clusters EKS que tenés en la región
<code>eksctl delete cluster --name X</code>	Elimina el cluster y todos sus recursos
<code>eksctl upgrade cluster --name X</code>	Actualiza la versión de Kubernetes del cluster

WORKER NODES

<code>eksctl get nodegroup --cluster X</code>	Lista los grupos de nodes del cluster
<code>eksctl scale nodegroup --cluster X --nodes 4</code>	Cambia la cantidad de nodes (escalar)
<code>eksctl create nodegroup --cluster X</code>	Agrega un grupo nuevo de nodes

DÓNDE CORREN TUS PODS: EC2 VS FARGATE

En EKS tenés dos opciones para los worker nodes:

EC2 (Managed Node Groups)

AWS crea instancias EC2 como worker nodes. Vos elegís el tipo de instancia

Fargate (Serverless)

No manejás ningún servidor. AWS crea un "mini servidor" efímero para cada Pod. No

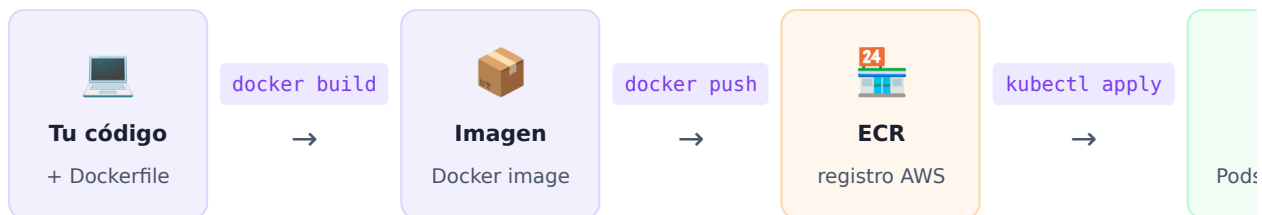
(t3.medium, m5.large, etc.) y cuántos. Tenés más control pero más que administrar.

Usá EC2 cuando: necesitás GPUs, tipos de instancia específicos, o querés controlar el costo con instancias reservadas o Spot.

elegís instancias, no parcheás nada. Pagás por Pod.

Usá Fargate cuando: querés simplicidad total, no querés pensar en servidores, y tu carga de trabajo es variable.

EL FLUJO COMPLETO: DE TU CÓDIGO A EKS



Paso a paso:

```
# 1. Construir la imagen Docker
docker build -t mi-backend:1.0 .

# 2. Etiquetar la imagen para ECR
docker tag mi-backend:1.0 123456789.dkr.ecr.us-east-1.amazonaws.com/mi-backend:1.0

# 3. Autenticarse en ECR
aws ecr get-login-password | docker login --username AWS --password-stdin 123456789.dkr.ecr.us-east-1.amazonaws.com

# 4. Subir la imagen a ECR
docker push 123456789.dkr.ecr.us-east-1.amazonaws.com/mi-backend:1.0

# 5. Desplegar a EKS
kubectl apply -f backend-deployment.yaml
kubectl apply -f backend-service.yaml
```

ECR = TU DOCKER HUB PRIVADO EN AWS

ECR (Elastic Container Registry) es donde guardás tus imágenes Docker dentro de AWS. Es privado, seguro, y se integra con EKS sin configuración extra. No tenés que usar Docker Hub si usás AWS.

INTEGRACIONES NATIVAS CON AWS

La ventaja de EKS frente a Kubernetes "genérico" son las integraciones automáticas con el ecosistema AWS:



IAM

Los permisos del cluster se gestionan con roles IAM. No hay que crear usuarios de K8s por separado.



VPC

Los Pods reciben IPs de tu VPC. La red de K8s y la red de AWS son la misma.



Load Balancer

Un Service tipo LoadBalancer crea automáticamente un ALB o NLB en AWS.



ECR

Registro privado de imágenes Docker. Se integra con EKS sin configuración extra.



CloudWatch

Logs y métricas de tus Pods directo a CloudWatch para monitoreo y alertas.



Secrets Manager

Inyecta secretos (contraseñas, API keys) a tus Pods sin guardarlos en el código.

EKS VS ECS: CUÁNDO USAR CADA UNO

↗ ECS — MÁS SIMPLE

- Servicio nativo de AWS, no es K8s
- Menos conceptos que aprender
- Sin costo de control plane (\$0)
- No portable a otros clouds
- Ideal si solo usás AWS y querés simplicidad

❄ EKS — MÁS POTENTE

- Kubernetes estándar — los YAML funcionan en cualquier cloud
- Ecosistema enorme de herramientas open source
- Control plane: ~\$72/mes
- Portable: migrás a GKE o AKS sin reescribir
- Ideal si ya usás K8s o necesitás portabilidad

COSTO REAL

CONTROL PLANE EKS

\$0.10/h

~\$72/mes. AWS gestiona el cluster, la HA, los backups. Sin esto no hay cluster.

WORKER NODES

EC2

Pagás las instancias normalmente. 2x t3.medium = ~\$60/mes. O usás Fargate y pagás por Pod.

EN LA PRÁCTICA

Un cluster EKS mínimo (control plane + 2 nodes t3.medium) cuesta ~**\$132/mes**. Para una startup chica, ECS puede ser más económico. Para una empresa con múltiples equipos desplegando muchos servicios, EKS se justifica rápidamente.

LA EVOLUCIÓN QUE VIERON EN EL CURSO

- **Sesión 2 → EC2 puro:** vos instalás, configurás y administrás el servidor
- **+ Docker:** tu app viaja empaquetada con su entorno, arranca en segundos
- **+ Kubernetes:** muchos contenedores coordinados automáticamente, self-healing, scaling
- **+ EKS:** Kubernetes sin mantener Kubernetes — AWS se encarga del cerebro

Cada paso sube un nivel de abstracción: menos trabajo operativo, más foco en el producto.