Kubernetes & Amazon EKS

Container Orchestration on AWS

Understanding Kubernetes, EKS vs ECS, and Architecture

AWS Training Series

What is Kubernetes?

Kubernetes (K8s)

An open-source container orchestration platform that automates deployment, scaling, and management of containerized applications.

Key Features:

- **Container Orchestration:** Manages containers across multiple hosts
- **Auto-scaling:** Automatically scales applications based on demand
- **Self-healing:** Restarts failed containers and replaces unhealthy nodes
- **Service Discovery:** Automatically discovers and connects services
- **Load Balancing:** Distributes traffic across healthy containers
- **Rolling Updates:** Updates applications without downtime

What is Amazon EKS?

Amazon Elastic Kubernetes Service (EKS)

A fully managed Kubernetes service that makes it easy to run Kubernetes on AWS without needing to install and operate your own Kubernetes control plane.

EKS Benefits:

- **Fully Managed:** AWS manages the Kubernetes control plane
- **High Availability:** Control plane runs across multiple AZs
- **Security:** Integrated with AWS IAM and VPC
- **Scalability:** Automatically scales control plane
- **AWS Integration:** Native integration with AWS services
- **Compliance:** SOC, PCI, ISO, and HIPAA compliant

Why EKS when we have ECS?

🎯 Use EKS When:

- You need Kubernetes-native features
- Multi-cloud or hybrid deployments
- Complex microservices architecture
- Existing Kubernetes expertise
- Need advanced networking (service mesh)
- Custom resource definitions (CRDs)

🚀 Use ECS When:

- Simple container workloads
- AWS-native applications
- Faster time to market
- Less operational overhead
- Tight AWS service integration
- Serverless containers (Fargate)

Key Insight: EKS provides Kubernetes flexibility and portability, while ECS offers simplicity and AWS-native integration.

EKS vs ECS Decision Matrix

EKS vs ECS: Detailed Comparison

Feature	Amazon EKS	Amazon ECS
Learning Curve	Steep (Kubernetes knowledge required)	Moderate (AWS-specific)
Portability	High (runs anywhere Kubernetes runs)	AWS-specific
Ecosystem	Rich Kubernetes ecosystem (Helm, operators)	AWS-native tools and services
Networking	Advanced (CNI plugins, service mesh)	Simple (AWS VPC integration)
Cost	$0.10/hour per cluster + worker nodes	No control plane cost + worker nodes
Scaling	Horizontal Pod Autoscaler, Cluster Autoscaler	Service Auto Scaling, Capacity Providers

Feature Comparison

Real-World Application Example

E-commerce Platform: "ShopFast"

🏪 ECS Implementation

- **Frontend:** React app on ECS Fargate
- **API Gateway:** ALB + ECS services
- **Microservices:** User, Product, Order services
- **Database:** RDS MySQL
- **Caching:** ElastiCache Redis

Best for: Simple deployment, tight AWS integration, faster development

⚙️ EKS Implementation

- **Frontend:** React pods with Ingress

- **API Gateway:** Istio service mesh
- **Microservices:** Kubernetes deployments
- **Database:** PostgreSQL operator
- **Monitoring:** Prometheus + Grafana

Best for: Complex workflows, multi-cloud strategy, advanced networking

> **Scenario:** ShopFast started with ECS for rapid deployment. As they grew globally and needed multi-cloud presence, they migrated critical services to EKS for portability and advanced features.

Kubernetes Architecture

Master Node (Control Plane)

API Server | **etcd** | **Scheduler** | **Controller Manager**

Worker Nodes

Node 1	**Node 2**	**Node 3**
kubelet	kubelet	kubelet
kube-proxy	kube-proxy	kube-proxy
Container Runtime	Container Runtime	Container Runtime

Control Plane Components

🎯 API Server

- Front-end for Kubernetes control plane
- Exposes Kubernetes API
- Validates and processes REST operations
- Gateway for all administrative tasks

💾 etcd

- Distributed key-value store
- Stores all cluster data
- Source of truth for cluster state
- Highly available and consistent

📅 Scheduler

- Assigns pods to nodes
- Considers resource requirements
- Evaluates constraints and policies
- Optimizes resource utilization

🎮 Controller Manager

- Runs controller processes
- Node Controller (monitors nodes)
- Replication Controller (maintains pod count)
- Service Account & Token Controllers

Worker Node Components

🤖 kubelet

- Primary node agent
- Communicates with API server
- Manages pod lifecycle
- Reports node and pod status
- Executes container operations

🌐 kube-proxy

- Network proxy on each node
- Maintains network rules
- Enables service abstraction
- Load balances traffic to pods
- Implements service discovery

📦 Container Runtime

- Runs containers (Docker, containerd, CRI-O)
- Pulls container images
- Manages container lifecycle
- Provides container isolation
- Handles resource allocation

In EKS: AWS manages the control plane components, while you manage the worker nodes (or use Fargate for serverless containers).

Worker Node Components

Key Kubernetes Objects

🏠 Pod

Smallest deployable unit containing one or more containers

🚀 Deployment

Manages replica sets and provides declarative updates

🔗 Service

Exposes pods as network services with stable endpoints

🌍 Ingress

Manages external access to services (HTTP/HTTPS routing)

📋 ConfigMap

Stores configuration data as key-value pairs

🔐 Secret

Stores sensitive data like passwords and API keys

Summary

Key Takeaways

- **Kubernetes** provides powerful container orchestration with flexibility and portability
- **Amazon EKS** offers managed Kubernetes with AWS integration and reduced operational overhead
- **Choose EKS** for complex microservices, multi-cloud needs, and Kubernetes ecosystem benefits
- **Choose ECS** for simpler workloads, faster deployment, and tight AWS integration
- **Architecture** separates control plane (managed by AWS in EKS) from worker nodes

Next Steps:

- Hands-on EKS cluster creation
- Deploy sample applications
- Explore Kubernetes networking
- Implement monitoring and logging

- Practice with Helm charts

Thank You!