Here is a **simple, detailed explanation of AWS EKS (Elastic Kubernetes Service)** with **all components**, **minute-level concepts**, and an **architecture diagram**.

✅ What is AWS EKS?

EKS (Elastic Kubernetes Service) is a **managed Kubernetes service** provided by AWS.
It helps you **run Kubernetes without managing the control plane**.

✅ Think of it like:

You want to deploy and manage containerized applications using **Kubernetes**, but don't want to install or manage the **Kubernetes master nodes, etcd, API server, etc.**
EKS does that **for you**, and you only manage the **worker nodes (EC2) or Fargate**.

🎯 Why Use EKS?

- No need to manage Kubernetes control plane
- Secure, scalable, and highly available
- Integrates with other AWS services (IAM, CloudWatch, ALB, etc.)
- Supports **EC2 or AWS Fargate** as worker nodes

🧱 Core Components of EKS

Component	Description
Control Plane	Managed by AWS. Runs Kubernetes API server, etcd, controller manager, scheduler
Worker Nodes	EC2 instances or AWS Fargate running your container workloads
Node Group	A group of EC2 instances managed by an Auto Scaling Group
VPC	Networking layer for your EKS cluster
IAM	Access and permissions for cluster and users
EKS Add-ons	Optional Kubernetes components like VPC CNI, CoreDNS, kube-proxy
ECR	Container image registry
Kubernetes Manifests	YAML files used to deploy applications
kubectl	CLI to interact with the Kubernetes cluster
aws-auth ConfigMap	Maps IAM roles/users to Kubernetes RBAC
Kubeconfig	Configuration file for accessing the cluster using `kubectl`
ALB/ELB/NLB	Load balancers to expose services

EKS Architecture Diagram (Text-Based)

```
┌──────────────────────────┐
│      AWS Control Plane   │   <-- Managed by AWS
│  - etcd                  │
│  - API Server            │
│  - Scheduler             │
└──────────────────────────┘
            │
            ▼
┌──────────────────────────────────┐
│            VPC (Network)         │
│  ┌──────────┐   ┌──────────────┐ │
│  │ Subnet A │   │  Subnet B    │ │
│  │(Private) │   │(Public/Priv) │ │
│  └──────────┘   └──────────────┘ │
│                                  │
│  ┌──────────────────────────┐    │
│  │     Worker Node Group    │    │
│  │(EC2 or Fargate instances)│    │
│  └──────────────────────────┘    │
│        ┌───┐  ┌───┐              │
│        │Pod│  │Pod│ <- App       │
│        └───┘  └───┘              │
└──────────────────────────────────┘
            ▲
            │
┌──────────────────────────┐
│  kubectl / eksctl        │
└──────────────────────────┘
```

🔍 Minute Concepts Explained (Simple Terms)

Concept	Explanation	Example
Pod	Smallest deployable unit in Kubernetes	A single container or a group of containers
Node	EC2 machine or Fargate that runs Pods	t3.medium EC2 running nginx
Node Group	Set of EC2s auto-managed as a group	Group of 3 EC2s running the same app
Service	Exposes Pods inside/outside the cluster	LoadBalancer type exposes to internet
Deployment	Manages replica Pods	Ensures 3 nginx Pods are always running
Ingress	Routes HTTP traffic to services	Domain routing like `api.example.com → service`
VPC CNI	Assigns IPs from VPC to Pods	Pod gets a real VPC IP like `10.0.2.1`
kube-proxy	Handles networking rules on node	Forwards traffic to correct Pod
CoreDNS	Internal DNS for service discovery	`my-service.default.svc.cluster.local`
IAM Roles for Service Accounts (IRSA)	Attach IAM to a Pod s[illegible]ely	Pod can access S3 via role

---

📦 Example EKS Workflow

1. **Create VPC** using Terraform or CloudFormation
2. **Create EKS cluster** using `eksctl` or Terraform
3. **Add Node Group** (EC2 or Fargate)
4. **Configure kubectl access** with `aws eks update-kubeconfig`
5. **Deploy App** using `kubectl apply -f deployment.yaml`
6. **Expose with LoadBalancer** using a Service
7. **Monitor** using CloudWatch, Prometheus, Grafana
8. **CI/CD** via Jenkins or GitHub Actions

💡 Real-World Example

You are deploying a Node.js app with MongoDB:

- EKS cluster with 3 EC2 nodes
- Node.js Pod with MongoDB Pod inside cluster
- Expose frontend using Ingress and ALB
- Store Docker image in ECR
- Jenkins pipeline automates build/test/deploy

🛡️ Security Components

Component	Role
IAM Roles	Control access to cluster resources
aws-auth ConfigMap	Maps IAM to Kubernetes users
Network Policies	Control traffic between Pods
Pod Security Policies (PSP)	Restrict pod capabilities
Secrets & ConfigMaps	Manage sensitive data securely

⚙️ EKS Deployment Tools

Tool	Purpose
eksctl	CLI for creating EKS easily
Terraform	Infrastructure as Code
Helm	Kubernetes package manager
kubectl	Main CLI to interact with EKS
Jenkins/GitHub Actions	CI/CD pipelines

📘 Summary in One Line

> **AWS EKS is a fully-managed Kubernetes service** where AWS handles the control plane, and you manage your containerized applications using EC2 or Fargate worker nodes.

Kubernetes Skills Covered

- KubeAPI server and scheduler
- Kubectl commands
- Kubernetes controllers
- Securing Kubernetes cluster
- Logging and monitoring clusters
- Troubleshooting cluster failures

Kubernetes Tools Covered

kubernetes HELM Prometheus Calico Kubectl