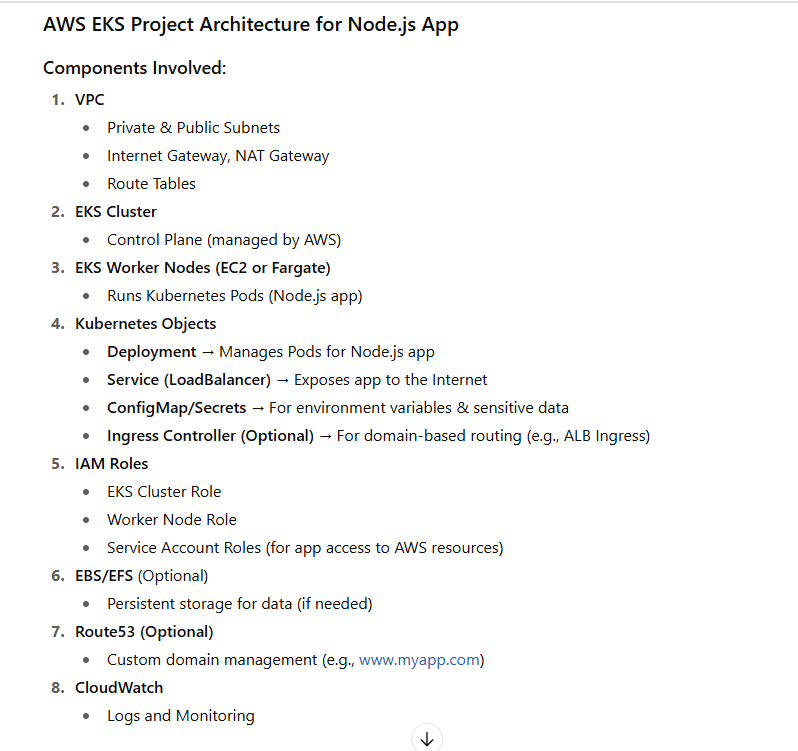
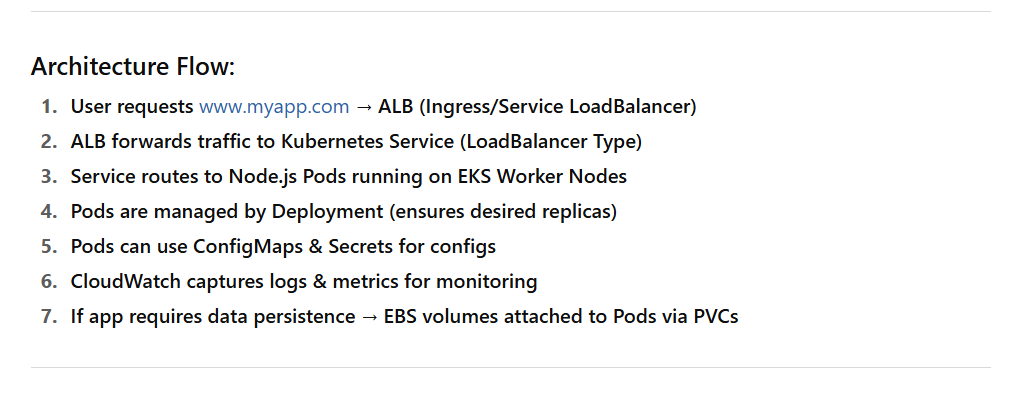
**Simple AWS EKS Architecture for deploying a Node.js application**:





A diagram of a structure flow

AI-generated content may be incorrect.

**step-by-step guide to create an EKS Cluster using AWS Console & AWS CLI** to deploy a **Node.js app** with the full architecture you outlined:

**🔧 PART 1: Networking Setup via AWS Console**

**Step 1: Create VPC**

* Go to **VPC Console → Your VPCs → Create VPC**
* Name: eks-vpc
* IPv4: e.g., 10.0.0.0/16
* Enable DNS hostnames: ✅ Yes

**Step 2: Create Subnets**

Create **4 Subnets**:

* 2 **Public** (e.g., 10.0.1.0/24, 10.0.2.0/24)
* 2 **Private** (e.g., 10.0.3.0/24, 10.0.4.0/24)
* Attach them to **2 different AZs** (e.g., us-east-1a, us-east-1b)

**Step 3: Internet Gateway & NAT Gateway**

* **Create Internet Gateway**, attach to VPC
* **Create Elastic IPs** for NAT Gateway
* **Create NAT Gateway** in one public subnet

**Step 4: Route Tables**

* **Public Route Table** → route 0.0.0.0/0 to Internet Gateway → associate to public subnets
* **Private Route Table** → route 0.0.0.0/0 to NAT Gateway → associate to private subnets

**☸️ PART 2: IAM Setup**

**Step 5: IAM Roles**

* **EKS Cluster Role**: AmazonEKSClusterPolicy
* **Node Group Role**:
  + AmazonEKSWorkerNodePolicy
  + AmazonEC2ContainerRegistryReadOnly
  + AmazonEKS\_CNI\_Policy

--------=-----------------------------------------------==========------------

**Step-6: Create IAM Node Role for EKS Worker Nodes**

This role allows EC2 instances (worker nodes) to connect to the EKS cluster and pull container images from ECR.

**✅ Option 1: Create Using AWS Console**

1. Go to **IAM → Roles → Create Role**
2. **Trusted Entity**: Select **AWS Service**
   * Use Case: **EC2**
   * Click **Next**
3. **Attach Permissions Policies**:  
   ✅ Select these 3 policies:
   * AmazonEKSWorkerNodePolicy
   * AmazonEKS\_CNI\_Policy
   * AmazonEC2ContainerRegistryReadOnly
4. **Name**: eks-node-group-role
5. **Tags** (Optional), then click **Create Role**
6. Copy the **Role ARN** — you’ll use it while creating the node group.

**✅ Option 2: Create Node Role via AWS CLI**

bash

CopyEdit

aws iam create-role \

--role-name eks-node-group-role \

--assume-role-policy-document file://trust-policy.json

**trust-policy.json**

json

CopyEdit

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Service": "ec2.amazonaws.com"

},

"Action": "sts:AssumeRole"

}

]

}

**✅ Attach Permissions to the Role**

bash

CopyEdit

aws iam attach-role-policy \

--role-name eks-node-group-role \

--policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy

aws iam attach-role-policy \

--role-name eks-node-group-role \

--policy-arn arn:aws:iam::aws:policy/AmazonEKS\_CNI\_Policy

aws iam attach-role-policy \

--role-name eks-node-group-role \

--policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly

**🔄 Reattempt Node Group Creation**

Now that the **Node Role is created**, retry your **node group creation** via:

* AWS Console: Select the role eks-node-group-role
* AWS CLI:

bash

CopyEdit

aws eks create-nodegroup \

--cluster-name my-eks-cluster \

--nodegroup-name my-node-group \

--scaling-config minSize=1,maxSize=3,desiredSize=2 \

--subnets subnet-abc subnet-def \

--instance-types t3.medium \

--ami-type AL2\_x86\_64 \

--node-role arn:aws:iam::<your-account-id>:role/eks-node-group-role \

--region us-east-1

---------------=---------------------------------------------------------------------

**☁️ PART 3: Create EKS Cluster (Console or AWS CLI)**

**Option 1: AWS Console**

1. Go to **EKS → Clusters → Create**
2. Name: my-eks-cluster
3. Kubernetes Version: Choose latest
4. Role: Choose EKS Role created earlier
5. Networking:
   * Choose the VPC and private subnets
   * Enable public endpoint (for now)
6. Create the cluster (takes 10-15 mins)

**Option 2: AWS CLI**

bash

CopyEdit

aws eks create-cluster \

--name my-eks-cluster \

--role-arn arn:aws:iam::<account-id>:role/eks-cluster-role \

--resources-vpc-config subnetIds=subnet-abc,subnet-def,securityGroupIds=sg-123 \

--region us-east-1

**⚙️ PART 4: Create Node Group (Console or CLI)**

**Option 1: Console**

1. Go to **EKS → Node groups → Add Node Group**
2. Name: node-group-1
3. Role: Node group role created earlier
4. Instance Type: t3.medium
5. Subnets: Choose **private subnets**
6. Create

**Option 2: AWS CLI**

bash

CopyEdit

aws eks create-nodegroup \

--cluster-name my-eks-cluster \

--nodegroup-name my-node-group \

--scaling-config minSize=1,maxSize=3,desiredSize=2 \

--subnets subnet-abc subnet-def \

--instance-types t3.medium \

--ami-type AL2\_x86\_64 \

--node-role arn:aws:iam::<account-id>:role/eks-node-group-role \

--region us-east-1

**PART 5: Connect with kubectl**

**Step 1: Update kubeconfig**

aws eks update-kubeconfig --region us-east-1 --name my-eks-cluster

**Step 2: Verify Connection**

kubectl get nodes

You should see nodes listed.

**🚀 PART 6: Deploy Node.js App to EKS**

**Step 1: Dockerize Node.js app**

Create Dockerfile:

Dockerfile

FROM node:18

WORKDIR /app

COPY . .

RUN npm install

CMD ["node", "index.js"]

**Step 2: Push to ECR**

**Step 3: Kubernetes YAML**

Create:

* deployment.yaml (Node.js deployment)
* service.yaml (LoadBalancer or ClusterIP)
* configmap.yaml (optional)
* secret.yaml (optional)

Apply:

bash

CopyEdit

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

**🔒 Optional: Ingress, Route53, EBS, CloudWatch**

**Ingress Controller (ALB)**

Use Helm to install ALB Ingress Controller and create Ingress resource.

**Route53**

Point a domain (e.g., www.myapp.com) to your ALB DNS.

**EBS/EFS**

Use PersistentVolume and PersistentVolumeClaim for stateful data.

**CloudWatch**

* Automatically enabled for EKS
* You can configure fluent-bit or cloudwatch-agent for more detailed logs.

**🧪 Optional: Monitoring with Prometheus + Grafana**

Let me know if you want this as the next step, and I’ll walk you through it.