KUBERNETES & TERRAFORM ARCHITECTURE

Complete Infrastructure Guide for DataMigrate AI

Author: Alexander Garcia Angus
Property of: OKO Investments

Generated: December 02, 2025

1. COMPLETE TECHNOLOGY STACK

Infrastructure Layer (Terraform):

* Amazon EKS - Kubernetes orchestration

* Amazon VPC - Network isolation

* Amazon RDS PostgreSQL - Relational database

* Amazon ElastiCache Redis - Caching & message broker

* Amazon ECR - Docker image registry

* AWS Secrets Manager - Secure credential storage

* Amazon S3 - Object storage & frontend hosting

* Amazon CloudFront - CDN for frontend

* AWS ALB - Application Load Balancer

* AWS CloudWatch - Monitoring & logging

Application Layer (Kubernetes):

* Go Backend - RESTful API (Gin framework)

* Vue.js 3 Frontend - Modern SPA (TypeScript)

* Python AI Service - LangGraph Agents

* Celery Workers - Background task processing

* Nginx Ingress - Kubernetes ingress controller

2. ARCHITECTURE OVERVIEW

The architecture follows a modern cloud-native design with the following flow:

Layer	Components	Purpose
DNS/CDN	Route53, CloudFront, WAF	DNS routing, edge caching, security
Load Balancing	AWS ALB	HTTPS termination, traffic distribution
Orchestration	Amazon EKS	Container orchestration (K8s 1.28)
Application	Go API, Python AI, Celery	Business logic, AI agents, background jobs
Data	RDS PostgreSQL, ElastiCache	Persistence, caching, message broker
Storage	S3, ECR	Artifacts, Docker images
Security	Secrets Manager, IAM IRSA	Credentials, service accounts

3. REQUEST FLOWS

Frontend Request (Vue.js):

User Browser -> CloudFront CDN (edge cache) -> S3 Static Website -> Downloads index.html, app.js, app.css -> User sees Vue.js app

API Request (Go Backend):

Vue.js App -> API call (fetch/axios) -> ALB -> Nginx Ingress -> Go API Pod (1 of 3 replicas) -> Redis cache check -> PostgreSQL if miss -> Response

Migration Request (AI Agents):

User initiates migration -> Go API receives request -> Creates Celery task -> Redis queue -> Celery Worker -> Python AI Service -> LangGraph multi-agent workflow:

1. Metadata Extraction Agent
2. Schema Analysis Agent
3. dbt Model Generator Agent
4. Validator Agent
5. Orchestrator Agent

-> Saves results to PostgreSQL -> Updates status -> Frontend polls for updates

4. KUBERNETES DEPLOYMENTS

Service	Replicas	Resources	Auto-scaling
Go API	3	512Mi RAM, 500m CPU	3-20 pods (70% CPU)
Python AI Service	2	1Gi RAM, 1000m CPU	2-10 pods
Celery Workers	5	1Gi RAM, 1000m CPU	2-20 pods (75% CPU)
Nginx Ingress	2	256Mi RAM, 250m CPU	Fixed

Key Kubernetes Features:

* Health checks (liveness + readiness probes)

* Horizontal Pod Autoscaler (HPA) based on CPU/memory

* External Secrets Operator for AWS Secrets Manager integration

* TLS termination with cert-manager (Let's Encrypt)

* Resource limits and requests for QoS

5. COST BREAKDOWN

Development Environment (~$250/month):

Service	Configuration	Monthly Cost
EKS Control Plane	1 cluster	$73.00
EKS Worker Nodes	2x t3.medium	$60.00
RDS PostgreSQL	db.t3.micro	$14.00
ElastiCache Redis	cache.t3.micro	$12.00
NAT Gateways	3x NAT	$32.40
ALB	Application LB	$16.00
S3 + CloudFront	Frontend	$10.00
Other (ECR, CW, SM)	Various	$40.00
TOTAL		**$258/month**

Production Environment (~$1,200-2,000/month):

Service	Configuration	Monthly Cost
EKS Control Plane	1 cluster	$73.00
EKS Worker Nodes	4-10x t3.large (avg 6)	$375.00

RDS PostgreSQL	db.t3.large Multi-AZ	$280.00
ElastiCache Redis	cache.m5.large (3 nodes)	$260.00
NAT + ALB + S3/CF	Production config	$150.00
Monitoring + WAF	CloudWatch + WAF	$130.00
TOTAL		**$1,331/month**

Cost Optimization Tips:

* Use Reserved Instances for RDS (40% savings)

* Use Spot Instances via Karpenter (70% savings)

* Implement auto-scaling (scale down during off-hours)

* Use VPC Endpoints to eliminate NAT costs

6. DEPLOYMENT WORKFLOW

Step 1: Build Docker Images

Build Go API, Python AI Service, and Celery Worker images. Tag and push to ECR.

Step 2: Deploy Infrastructure (Terraform)

cd terraform/environments/dev && terraform init && terraform plan && terraform apply

Step 3: Configure kubectl

aws eks update-kubeconfig --region us-east-1 --name datamigrate-ai-dev-eks

Step 4: Deploy Kubernetes Resources

kubectl create namespace datamigrate-ai && kubectl apply -f k8s/external-secrets.yaml && kubectl apply -f k8s/deployments/ && kubectl apply -f k8s/ingress.yaml

Step 5: Deploy Frontend

cd frontend && npm run build && aws s3 sync dist/ s3://datamigrate-ai-frontend --delete && aws cloudfront create-invalidation --distribution-id --paths '/*'

7. TECHNOLOGY STACK SUMMARY

Category	Technology	Purpose
IaC	Terraform	Infrastructure as Code
Orchestration	Kubernetes (EKS)	Container orchestration
Backend	Go (Gin) + Python	REST API + AI agents
Frontend	Vue.js 3 + TypeScript	SPA framework
AI Framework	LangGraph + LangChain	Multi-agent system
Task Queue	Celery + Redis	Background jobs
Database	PostgreSQL 15 (RDS)	Relational data
Cache	Redis (ElastiCache)	Caching + broker
CDN	CloudFront	Frontend delivery
CI/CD	GitHub Actions	Automated deployment

This is an enterprise-grade, production-ready architecture designed for OKO Investments!
