Development Infrastructure

McMillan Project Documentation

September 2, 2025

Overview

The development infrastructure is designed to support a modular, containerized application that integrates backend services, AI-based document processing, a relational database, and frontend applications. It leverages **AWS Cloud resources** for scalability, cost efficiency, and reliability. **Nginx acts as a reverse proxy** routing traffic to appropriate services.

1 Core Components

1.1 Nginx (Reverse Proxy)

- **Single entry point** for all external traffic (port 80/443)
- Routes requests to appropriate backend services
- Handles SSL termination and load balancing
- Serves static files and handles CORS policies
- Configuration mounted from host: `./nginx/nginx.conf:/etc/nginx/nginx.conf`

Routing Configuration:

- `/api/*` → FastAPI Backend (port 8000)
- `/invoiceai/*` → InvoiceAI Service (port 8001)
- `/pgadmin/*` → pgAdmin (port 5050)
- `/app1/*` → Frontend App 1 (S3 redirect)
- `/app2/*` → Frontend App 2 (S3 redirect)

1.2 Backend (FastAPI)

- Hosted in Docker container (`app` service)
- **No longer directly exposed** - accessed via Nginx proxy
- Handles REST API requests, authentication, and communication with database
- Uses environment variables and `.env` file for configuration
- Logs are persisted to host machine (`./app/logs:/app/logs`)

1.3 InvoiceAI Service

- Custom container (`consulttechencraft/invoiceai`)
- **Accessed through Nginx** at `/invoiceai` path
- Communicates with **OpenAI API** for AI-driven document analysis
- Handles OCR/text processing before results are sent to Postgres
- Runs independently but shares the same **Docker network** for inter-service communication

1.4 Database (Postgres)

- Dockerized Postgres 15
- Persistent storage via Docker volume `db_data`
- **Internal access only** - not exposed through Nginx
- Health checks ensure DB is running before dependent services start

1.5 pgAdmin

- UI management tool for Postgres
- **Accessed through Nginx** at `/pgadmin` path
- Uses admin credentials defined in `.env`
- Enhanced security through proxy configuration

1.6 Frontend Applications

- Two static web applications (e.g., React/Angular/Vue builds)
- Hosted in **AWS S3 buckets** with static website hosting enabled
- **Proxied through Nginx** for unified domain access
- Delivered via **S3 public URLs with Nginx caching**

2 Service Architecture

Updated Container Configuration

- **Nginx Service**: Alpine-based image with HTTP (80) and HTTPS (443) ports
- **Configuration Mounting**: Nginx config and SSL certificates mounted from host
- **Service Dependencies**: Nginx depends on app, invoiceai, and pgladmin services
- **Internal Network**: All services communicate via mcmillan-net Docker network
- **Security Model**: Backend services no longer expose external ports

3 AWS Infrastructure

3.1 EC2 Instance

- **t3.medium** (2 vCPUs, 4 GiB RAM)
- Runs Docker + Docker Compose
- Containers: **Nginx**, FastAPI backend, InvoiceAI, Postgres, pgAdmin
- Cost optimized: Runs ∼10 hrs/day (∼$12.5/month)
- **Single public port** (80/443) improves security

3.2 EBS (Elastic Block Store)

- 20 GiB gp3 volume for EC2 instance storage
- Persists database, application logs, and Nginx configuration

3.3 S3 Buckets

- **10 GiB S3 storage** for document data
- **2 S3 buckets** for frontend apps hosting
- **Nginx caches** S3 content for improved performance
- Cost-effective and highly available

4 Networking & Security

- **VPC** with private Docker bridge network (`mcmillan-net`) for service communication
- **Nginx as security gateway**:
  - Single entry point (ports 80/443)
  - Rate limiting and DDoS protection

  - Request filtering and validation
  - SSL/TLS termination

- **Internal services** no longer directly exposed:
  - Backend API (internal port 8000)
  - InvoiceAI service (internal port 8001)
  - pgAdmin (internal port 5050)
  - Postgres (internal port 5432)
- Security Groups now only allow HTTP/HTTPS traffic
- **IAM roles & policies** manage S3 access

5 Monitoring & Logging

- **CloudWatch** for EC2 instance metrics (CPU, memory, disk usage)
- **Nginx access logs** for traffic analysis and security monitoring
- **FastAPI + InvoiceAI logs** persisted in `./app/logs`
- **Postgres logs** available via container logs
- **Centralized logging** through Nginx for all HTTP requests

6 $ Cost Summary (Monthly)

Component	**Usage**	**Est. Cost**
EC2 (t3.medium, 10 hrs/day)	Compute	$12.48
EBS (20 GiB gp3)	Storage	$1.60
S3 (10 GiB backend data)	Storage	$0.23
S3 (2 frontend apps + traffic)	Hosting & transfer	$0.85
Nginx (included in EC2)	Reverse proxy	$0.00
Total		**$15.2**

Table 1: Monthly cost breakdown for development infrastructure

7 Scalability Path

Short Term

t3.medium is sufficient for current requirements. **Nginx provides excellent performance optimization** through caching and compression.

If Traffic Increases

- Upgrade to **t3.large (8 GiB RAM)**
- **Nginx load balancing** to multiple backend instances
- Offload Postgres to **Amazon RDS** for managed performance
- Add **CloudFront** in front of Nginx for global CDN

If AI Workload Grows

Switch to containerized GPU inference (ECS/EKS) or continue relying on OpenAI API. **Nginx can route** AI requests to multiple InvoiceAI instances.

Summary

This setup ensures a **enhanced development environment**:

- Low cost (∼$15/month) - **no additional cost for Nginx**
- **Improved security** through single entry point
- **Better performance** with caching and compression
- Containers provide isolation & easy deployment
- AWS services (EC2, S3, EBS) provide scalability & persistence
- **Unified routing** for all services and frontend apps