DEPLOYMENT MODEL

STOCKFELLOW

DEVOPPS

BRIGHTBYTE ENTERPRISES

DEMO 4

Name and Surname	**Student Number**
*Tinotenda Chirozvi	22547747
Diyaana Jadwat	23637252
Dean Ramsey	22599012
Naazneen Khan	22527533
Lubabalo Tshikila	22644106

Contents

1 Backend Deployment Diagram

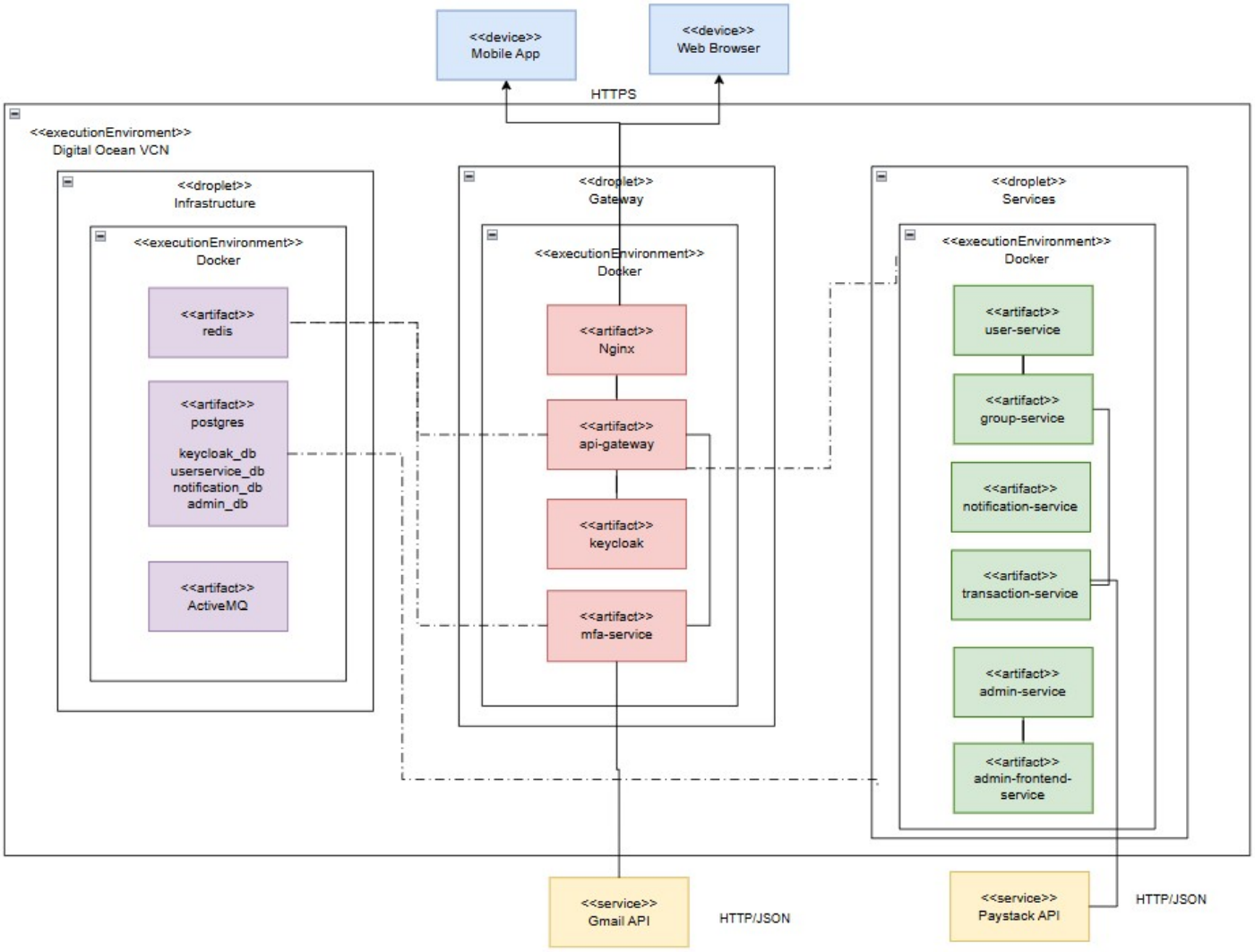

Figure 1: Backend Deployment Diagram

2 Deployment Environment

The StockFellow fintech backend will be deployed as a containerized Java Spring Boot microservices architecture on Digital Ocean Droplets. The system uses Docker Compose for orchestration and is cost-effective with very capable hardware, by splitting the services between 3 droplets.

2.1 Target Platform

- **Cloud Provider:** Digital Ocean (DO)
- **Instance Type:** 3x Droplets on a VCN
- **Specifications:** 4x 4GB RAM, 4 OCPUs, 25GB Storage

- **Operating System:** Ubuntu 22.04 LTS
- **Container Runtime:** Docker with Docker Compose

2.2 Reasons for Digital Ocean Selection

- Low/No cost with generous Free Tier
- Sufficient resources for all microservices and databases
- VCN always for fast (free) communication between droplets

2.3 Deployment Pattern

The system implements a multi-server containerized microservices architecture with:

- **NGINX Reverse Proxy:** SSL termination and load balancing
- **Java Spring Boot Services:** Business logic microservices
- **PostgreSQL Databases:** Relational data storage
- **Redis Cache:** Session management and caching

3 Service Architecture Overview

3.1 Client Access Layer

- Mobile Apps: React Native/Flutter applications
- Web Admin Panel: Browser-based administration
- External Access: HTTPS through domain name or public IP

3.2 Proxy Layer

- NGINX: Port 80/443 (SSL termination, reverse proxy) Running in front of te API-Gateway which then direts traffic to services

3.3 Application Layer (Java Spring Boot)

- API Gateway: Port 3000 (Request routing, authentication)
- User Service: Port 4020 (User management, profiles)
- Group Service: Port 4040 (Investment groups, communities)
- Transaction Service: Port 4080 (Payment processing, Paystack integration)
- MFA Service: Port 8087 (Multi-factor authentication)
- Notification Service: Port 4050 (Push notifications, messaging)
- Keycloak: Port 8080 (Identity and access management)
- Admin Service: Port 4060 (Admin dashboard and management)

3.4 Database Layer

For PostgreSQL, there is a single Postgres Service that hosts multiple databases. The group service uses Mongo Atlas to provide a NoSQL database for group storage.

- Keycloak PostgreSQL: Port 5432 (Identity data)
- User Service PostgreSQL: Port 5432 (User profiles, accounts)
- Notification PostgreSQL: Port 5432 (Notification history)
- Group MongoDB: Port 5433 (Group data)
- Redis Cache: Port 6379 (Sessions, cache)
- ActiveMQ: Port 61616 (Message broker)

3.5 External Services

- Gmail SMTP: Email delivery for MFA
- Paystack API: Payment processing

3.6 Service Technology Table

Service	**Technology**	**Port / Purpose**
postgres	PostgreSQL 15	5432 (Keycloak config + data)
redis	Redis 7	6379 (Cache, sessions)
active-mq	ActiveMQ Artemis	61616 (Message brokering)
nginx-proxy	NGINX	80, 443 (Reverse proxy, SSL termination)
api-gateway	Spring Boot	3000 (API routing, authentication)
user-service	Spring Boot	4000 (User management)
group-service	Spring Boot	4040 (Group operations)
transaction-service	Spring Boot	4080 (Payment processing)
mfa-service	Spring Boot	8087 (Multi-factor authentication)
notification-service	Spring Boot	4050 (Notifications)
keycloak	Keycloak	8180 (Identity management)
admin-service	Spring Boot	4060 (Admin dashboard)
admin-frontend	NodeJS	4070 (Admin dashboard UI)

Table 1: Service Technology and Ports

4 Container Architecture

4.1 Droplet 1: Infrastructure Services

- **Database Containers**
  - PostgreSQL (15-alpine) - Multi-database instance serving keycloak_db, userservice_db, and notification_db

  - Redis (7-alpine) - Session cache and data store with 768MB memory limit
  - ActiveMQ Artemis - Message broker for asynchronous communication

- **Monitoring Containers**
  - Glances Resource Monitor - Real-time system monitoring on port 61208

4.2 Droplet 2: Gateway Services

- **Authentication Containers**
  - Keycloak - Identity and access management with realm imports
  - Nginx Reverse Proxy - SSL termination and load balancing

- **Application Containers**
  - API Gateway - Spring Cloud Gateway for service routing and authentication
  - MFA Service - Multi-factor authentication with SendGrid integration

4.3 Droplet 3: Business Services

- **Application Containers**
  - User Service - User management with PostgreSQL persistence
  - Group Service - Investment group management with MongoDB
  - Transaction Service - Payment processing with Paystack integration
  - Notification Service - Event-driven notifications with ActiveMQ
  - Admin Service - Admin Management with RBAC through Keyclaok
  - Admin Front-end - UI for the admin dashboard

5 Network Architecture

5.1 Security Model

- External Access: Only NGINX (ports 80, 443) exposed to internet
- Internal Network: All services communicate via Docker bridge network and inter-droplet communication is facilitated by the Digital Ocean VCN
- SSL/TLS: NGINX handles SSL termination
- Authentication: JWT tokens via Keycloak for all API access
- Role Based Access Control: On Admin users with the Admin role can access the Admin Dashboard

5.2 Communication Flow

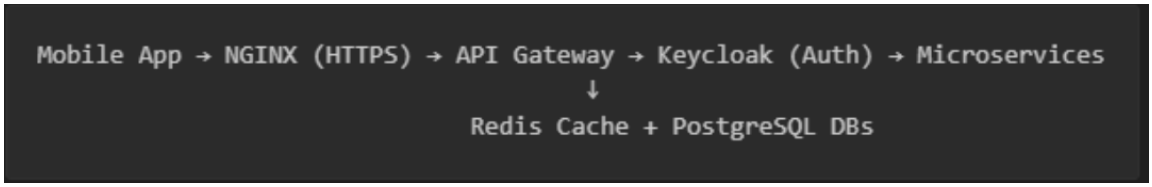

Figure 2: Communication Flow

6 Deployment Process

6.1 Configuration

- Environment Variables: All configuration via .env file
- Secrets: Database passwords, API keys in environment variables
- External URLs: Domain-based URLs for production
- JVM Settings: Optimized for 3x 4GB RAM droplet allocation

6.2 Workflow

- CI Pipeline: Runs all tests and linting
- CD Pipeline: Deploys generated Docker artifacts to server's Docker instance

7 Resource Allocation

7.1 Memory Distribution

- Java Services: 10GB (6 services × 1.5GB each)
- Keycloak: 2GB
- PostgreSQL: 1.5GB (3 instances)
- Redis + ActiveMQ: 0.5GB
- NGINX: 0.1GB
- System/OS: 6GB
- Buffer: 4GB

8 Monitoring and Maintenance

8.1 Health Monitoring

- Health Endpoints: All Spring Boot services expose `/actuator/health`
- Database Health: `pg_isready` for PostgreSQL instances
- System Monitoring: `monitor.sh` script for resource usage

8.2 Logging

- Application Logs: Docker container logs
- Access Logs: NGINX request logging
- Error Tracking: Centralized via Docker logs

9 Security Considerations

9.1 Network Security

- Firewall: Only ports 22, 80, 443 open to internet
- Internal Communication: Services isolated in Docker network
- SSL/TLS: All external communication encrypted
- SSH: RSA keys and passphrases are used to secure SSH access to droplets

9.2 Application Security

- OAuth 2.0: Authentication via Keycloak
- JWT Tokens: Secure service-to-service communication
- Database Security: Isolated databases per service
- Secret Management: Environment variable based
- Tokenized Payment Details: via Paystack Authorizations