AfricanMarket Deployment Guide

This document provides comprehensive instructions for deploying the AfricanMarket application to various environments.

🚀 Deployment Overview

The AfricanMarket application supports multiple deployment strategies:
- **Development**: Local development environment
- **Staging**: Testing environment that mirrors production
- **Production**: Live production environment

🔧 Prerequisites

System Requirements

- **Node.js**: v18 or higher
- **PostgreSQL**: v13 or higher
- **Redis**: v6 or higher (optional, for caching)
- **Memory**: Minimum 2GB RAM
- **Storage**: Minimum 10GB available space

Required Tools

- `node` and `npm` / `yarn`
- `git`
- `docker` (optional)
- `pm2` (for process management)

🌍 Environment Configuration

1. Environment Variables

Create environment-specific files:

`.env.development`

```
# Database
DATABASE_URL="postgresql://postgres:password@localhost:5432/africanmarket_dev"

# NextAuth
NEXTAUTH_URL="http://localhost:3000"
NEXTAUTH_SECRET="development-secret-key"

# Push Notifications
VAPID_PUBLIC_KEY="BM8ZaJGkVIIpK3F9XJjGKbYhU4fIVnJIlNkZtmHgN3RjQmHsT3kJ9PZEqLxVfG-
mJ4kH3QxV9wMsZmNhY5sK2F7E"
VAPID_PRIVATE_KEY="gN8F3kJ9PZEqLxVfGmJ4kH3QxV9wMsZmNhY5sK2F7E"

# Application
NEXT_PUBLIC_BASE_URL="http://localhost:3000"
NODE_ENV="development"
PORT=3000

# Optional: Redis for caching
REDIS_URL="redis://localhost:6379"
```

`.env.staging`

```
# Database
DATABASE_URL="postgresql://africanmarket:password@staging-db:5432/africanmar-
ket_staging"

# NextAuth
NEXTAUTH_URL="https://staging.africanmarket.com"
NEXTAUTH_SECRET="staging-secret-key-generate-new-one"

# Push Notifications
VAPID_PUBLIC_KEY="your-staging-vapid-public-key"
VAPID_PRIVATE_KEY="your-staging-vapid-private-key"

# Application
NEXT_PUBLIC_BASE_URL="https://staging.africanmarket.com"
NODE_ENV="production"
PORT=3000

# Redis
REDIS_URL="redis://staging-redis:6379"
```

`.env.production`

```
# Database
DATABASE_URL="postgresql://africanmarket:secure-password@production-db:5432/africanmar-
ket"

# NextAuth
NEXTAUTH_URL="https://africanmarket.com"
NEXTAUTH_SECRET="production-secret-key-very-secure"

# Push Notifications
VAPID_PUBLIC_KEY="your-production-vapid-public-key"
VAPID_PRIVATE_KEY="your-production-vapid-private-key"

# Application
NEXT_PUBLIC_BASE_URL="https://africanmarket.com"
NODE_ENV="production"
PORT=3000

# Redis
REDIS_URL="redis://production-redis:6379"

# Optional: External services
SENTRY_DSN="your-sentry-dsn"
LOGFLARE_API_KEY="your-logflare-api-key"
```

2. Database Setup

PostgreSQL Configuration

```
-- Create database
CREATE DATABASE africanmarket;

-- Create user
CREATE USER africanmarket WITH PASSWORD 'secure-password';

-- Grant privileges
GRANT ALL PRIVILEGES ON DATABASE africanmarket TO africanmarket;
```

Redis Configuration (Optional)

```
# Start Redis server
redis-server

# Configure Redis for production
# Edit /etc/redis/redis.conf
bind 127.0.0.1
port 6379
maxmemory 256mb
maxmemory-policy allkeys-lru
```

🚢 Deployment Methods

Method 1: Automated Deployment Script

The recommended deployment method using the provided script:

```
# Deploy to staging
./scripts/deploy.sh staging

# Deploy to production
./scripts/deploy.sh production 3000 production

# Check deployment status
./scripts/deploy.sh health
```

Method 2: Manual Deployment

Step 1: Prepare the Environment

```
# Clone repository
git clone <repository-url>
cd africanmarket/app

# Install dependencies
yarn install --frozen-lockfile

# Set up environment
cp .env.production .env
```

Step 2: Database Setup

```
# Generate Prisma client
npx prisma generate

# Run migrations
npx prisma migrate deploy

# Seed database (if needed)
npx prisma db seed
```

Step 3: Build Application

```
# Build the application
yarn build

# Verify build
yarn start --port 3000
```

Step 4: Process Management

```
# Install PM2 globally
npm install -g pm2

# Create PM2 ecosystem file
cat > ecosystem.config.js << EOF
module.exports = {
  apps: [{
    name: 'africanmarket',
    script: 'yarn',
    args: 'start',
    cwd: '/path/to/africanmarket/app',
    env: {
      NODE_ENV: 'production',
      PORT: 3000
    },
    instances: 'max',
    exec_mode: 'cluster',
    watch: false,
    max_memory_restart: '1G',
    error_file: './logs/err.log',
    out_file: './logs/out.log',
    log_file: './logs/combined.log'
  }]
}
EOF

# Start application
pm2 start ecosystem.config.js
pm2 save
pm2 startup
```

Method 3: Docker Deployment

Step 1: Create Dockerfile

```dockerfile
FROM node:18-alpine

WORKDIR /app

# Copy package files
COPY package.json yarn.lock ./

# Install dependencies
RUN yarn install --frozen-lockfile

# Copy application code
COPY . .

# Generate Prisma client
RUN npx prisma generate

# Build application
RUN yarn build

# Expose port
EXPOSE 3000

# Start application
CMD ["yarn", "start"]
```

Step 2: Create docker-compose.yml

```yaml
version: '3.8'

services:
  app:
    build: .
    ports:
      - "3000:3000"
    environment:
      - NODE_ENV=production
      - DATABASE_URL=postgresql://africanmarket:password@db:5432/africanmarket
      - NEXTAUTH_URL=https://your-domain.com
      - NEXTAUTH_SECRET=your-secret
      - REDIS_URL=redis://redis:6379
    depends_on:
      - db
      - redis
    volumes:
      - ./uploads:/app/uploads
      - ./logs:/app/logs

  db:
    image: postgres:13
    environment:
      - POSTGRES_USER=africanmarket
      - POSTGRES_PASSWORD=password
      - POSTGRES_DB=africanmarket
    volumes:
      - postgres_data:/var/lib/postgresql/data
    ports:
      - "5432:5432"

  redis:
    image: redis:6-alpine
    ports:
      - "6379:6379"
    volumes:
      - redis_data:/data

volumes:
  postgres_data:
  redis_data:
```

Step 3: Deploy with Docker

```bash
# Build and start services
docker-compose up -d

# Run migrations
docker-compose exec app npx prisma migrate deploy

# Check logs
docker-compose logs -f app
```

🔍 Health Checks & Monitoring

Health Check Endpoints

```
# Basic health check
curl http://localhost:3000/api/health

# Detailed health check
curl http://localhost:3000/api/health?detailed=true

# System metrics (requires admin auth)
curl -H "Authorization: Bearer <admin-token>" http://localhost:3000/api/metrics
```

Monitoring Setup

1. Application Monitoring

```
# Install monitoring tools
npm install -g pm2-logrotate

# Configure log rotation
pm2 install pm2-logrotate

# Monitor application
pm2 monit
```

2. Database Monitoring

```
-- Monitor database performance
SELECT * FROM pg_stat_activity WHERE state = 'active';
SELECT * FROM pg_stat_database WHERE datname = 'africanmarket';
```

3. System Monitoring

```
# Monitor system resources
htop
iotop
df -h
free -m

# Monitor application logs
tail -f logs/combined.log
```

🛡️ Security Considerations

SSL/TLS Configuration

```
# Using Let's Encrypt with Certbot
sudo certbot --nginx -d africanmarket.com -d www.africanmarket.com
```

Firewall Configuration

```
# Configure UFW
sudo ufw allow ssh
sudo ufw allow 80
sudo ufw allow 443
sudo ufw allow 3000
sudo ufw enable
```

Environment Security

```
# Secure environment files
chmod 600 .env*
chown app:app .env*

# Use secrets management
# Store sensitive data in environment variables or secret management services
```

🔧 Troubleshooting

Common Issues

Database Connection Issues

```
# Check database status
systemctl status postgresql

# Test connection
psql -U africanmarket -h localhost -d africanmarket

# Check connection pool
npx prisma db pull
```

Memory Issues

```
# Check memory usage
free -m
ps aux --sort=-%mem | head

# Increase swap if needed
sudo fallocate -l 2G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
```

Port Conflicts

```
# Check port usage
netstat -tlnp | grep :3000
lsof -i :3000

# Kill process using port
sudo kill -9 <PID>
```

Log Analysis

```
# Application logs
tail -f logs/combined.log

# Error logs
tail -f logs/err.log

# Database logs
sudo tail -f /var/log/postgresql/postgresql-13-main.log

# System logs
sudo journalctl -u africanmarket -f
```

📊 Performance Optimization

Database Optimization

```
-- Create indexes for better performance
CREATE INDEX idx_users_email ON users(email);
CREATE INDEX idx_rides_status ON rides(status);
CREATE INDEX idx_rides_customer_id ON rides(customer_id);
```

Application Optimization

```
# Enable compression
# Add to next.config.js
const nextConfig = {
  compress: true,
  experimental: {
    optimizeCss: true,
    optimizeImages: true
  }
}
```

Caching Configuration

```
// Configure Redis caching
const redis = require('redis');
const client = redis.createClient({
  url: process.env.REDIS_URL
});
```

🔄 Rollback Procedures

Automated Rollback

```
# Rollback using deployment script
./scripts/deploy.sh rollback
```

Manual Rollback

```
# Stop current application
pm2 stop africanmarket

# Restore from backup
cp -r backup/latest/* ./

# Run previous migrations
npx prisma migrate reset
npx prisma migrate deploy

# Restart application
pm2 start africanmarket
```

📋 Maintenance Tasks

Regular Maintenance

```
# Update dependencies
yarn upgrade

# Clean up logs
find logs -name "*.log" -mtime +30 -delete

# Database maintenance
npx prisma db push
npx prisma db seed

# System updates
sudo apt update && sudo apt upgrade
```

Backup Procedures

```
# Database backup
pg_dump -h localhost -U africanmarket africanmarket > backup/db-$(date +%Y%m%d).sql

# Application backup
tar -czf backup/app-$(date +%Y%m%d).tar.gz .

# Automated backup script
cat > backup.sh << 'EOF'
#!/bin/bash
BACKUP_DIR="/backups"
DATE=$(date +%Y%m%d_%H%M%S)
pg_dump $DATABASE_URL > $BACKUP_DIR/db-$DATE.sql
tar -czf $BACKUP_DIR/app-$DATE.tar.gz /app
find $BACKUP_DIR -name "*.sql" -mtime +7 -delete
find $BACKUP_DIR -name "*.tar.gz" -mtime +7 -delete
EOF

chmod +x backup.sh
crontab -e
# Add: 0 2 * * * /path/to/backup.sh
```

🚨 Emergency Procedures

Application Down

```
# Quick restart
pm2 restart africanmarket

# Check logs
pm2 logs africanmarket

# Rollback if needed
./scripts/deploy.sh rollback
```

Database Issues

```
# Restart database
sudo systemctl restart postgresql

# Check database status
sudo systemctl status postgresql

# Restore from backup
psql -U africanmarket -d africanmarket < backup/latest.sql
```

High Load

```
# Scale application
pm2 scale africanmarket +2

# Check resource usage
htop
iotop

# Enable caching
# Check Redis status
redis-cli ping
```

This deployment guide provides comprehensive instructions for deploying the AfricanMarket application in various environments. Follow the appropriate method based on your infrastructure and requirements.