🐳 EscalaFin - Despliegue con Docker en EasyPanel

📋 Guía Completa para Despliegue con Dockerfile

Esta guía te llevará paso a paso para desplegar EscalaFin en EasyPanel usando Docker con configuración optimizada.

✅ Pre-requisitos

- [ ] Cuenta activa en EasyPanel
- [ ] Repositorio GitHub con el código actualizado
- [ ] Dockerfile y docker-compose.yml (ya incluidos)
- [ ] Credenciales de producción listas

🗂️ Archivos Docker Incluidos

📄 Dockerfile

```
FROM node:18-alpine AS base
# Configuración multi-stage optimizada para producción
# - Stage deps: Instala dependencias
# - Stage builder: Compila la aplicación
# - Stage runner: Imagen final optimizada
```

📄 .dockerignore

```
# Excluye archivos innecesarios del build context
node_modules/
.git/
*.md
.env*
```

📄 docker-compose.yml

```
# Para desarrollo local con PostgreSQL y Redis
services:
  app: # Aplicación Next.js
  db:  # PostgreSQL 14
  redis: # Redis para caché (opcional)
```

🏗️ Paso 1: Configurar en EasyPanel

1.1 Crear Nueva Aplicación

1. **Accede a tu dashboard de EasyPanel**
2. **Crear nuevo proyecto/servicio**
   - Nombre: `escalafin-app`
   - Tipo: `App`
   - Source: `GitHub Repository`
3. **Configurar repositorio**
   - Repositorio: `tu-usuario/escalafin-mvp`
   - Branch: `main`
   - Dockerfile Path: `Dockerfile` (raíz del proyecto)

1.2 Configurar Build Settings

```
# Build Configuration
Build Context: /
Dockerfile: Dockerfile
Port: 3000

# Build Command (opcional, Docker lo maneja)
Build Command: (dejar vacío)

# Start Command (Docker lo maneja)
Start Command: (dejar vacío)
```

🗄️ Paso 2: Configurar PostgreSQL

2.1 Crear Base de Datos

1. **Agregar servicio de base de datos**
   - Tipo: `PostgreSQL`
   - Versión: `14` o superior
   - Nombre: `escalafin-db`
2. **Configuración de la DB**
   `Database Name: escalafin`
   `Username: escalafin_user`
   `Password: [generar password seguro]`
   `Port: 5432`
   `Storage: 10GB (mínimo)`

2.2 Configurar Variables de Entorno

En la aplicación EasyPanel, añadir estas variables:

```
# === APLICACIÓN ===
NODE_ENV=production
NEXT_TELEMETRY_DISABLED=1

# === BASE DE DATOS ===
# EasyPanel proporcionará la URL interna:
DATABASE_URL=postgresql://escalafin_user:PASSWORD@escalafin-db:5432/escalafin

# === AUTENTICACIÓN ===
NEXTAUTH_URL=https://tu-app.easypanel.app
NEXTAUTH_SECRET=tu_secret_super_seguro_minimo_32_caracteres

# === OPENPAY PRODUCCIÓN ===
OPENPAY_MERCHANT_ID=tu_merchant_id_real
OPENPAY_PRIVATE_KEY=tu_private_key_real
OPENPAY_PUBLIC_KEY=tu_public_key_real
OPENPAY_BASE_URL=https://api.openpay.mx/v1

# === AWS S3 PRODUCCIÓN ===
AWS_ACCESS_KEY_ID=tu_access_key_produccion
AWS_SECRET_ACCESS_KEY=tu_secret_key_produccion
AWS_BUCKET_NAME=escalafin-production
AWS_REGION=us-east-1
AWS_FOLDER_PREFIX=production/

# === WHATSAPP PRODUCCIÓN ===
EVOLUTION_API_URL=https://tu-evolution-api-produccion.com
EVOLUTION_API_TOKEN=tu_token_evolution_produccion
EVOLUTION_INSTANCE_NAME=escalafin-production

# === CONFIGURACIÓN ADICIONAL ===
RATE_LIMIT_MAX=100
RATE_LIMIT_WINDOW=900000
```

🚀 Paso 3: Primer Despliegue

3.1 Ejecutar Deploy

1. **Iniciar build**
   - Click en `Deploy` en EasyPanel
   - Monitorear logs de build en tiempo real
   - Verificar que todos los stages completaron:

   ```
   ✅ [deps] Dependencies installed
   ✅ [builder] Prisma client generated
   ✅ [builder] Next.js build completed
   ✅ [runner] Production image created
   ```

2. **Verificar el container**
   - Status: `Running`
   - Health Check: `Healthy`
   - Logs sin errores críticos

3.2 Inicializar Base de Datos

Ejecutar una sola vez tras el primer despliegue:

```
# Opción 1: Via terminal de EasyPanel
docker exec -it escalafin-app npx prisma db push
docker exec -it escalafin-app npx prisma db seed

# Opción 2: Via API endpoint (si está configurado)
curl -X POST https://tu-app.easypanel.app/api/admin/init-db
```

3.3 Verificar Funcionamiento

1. **Health Check**
   `bash`
   `curl https://tu-app.easypanel.app/api/health`
   `# Debe retornar: {"status":"healthy","database":"connected"}`

2. **Login de prueba**
   - URL: `https://tu-app.easypanel.app/login`
   - Usuario: `admin@escalafin.com`
   - Contraseña: `Admin123!`

🌐 Paso 4: Configurar Dominio Personalizado

4.1 Agregar Dominio en EasyPanel

1. **App Settings → Domains**
   - Dominio: `escalafin.tudominio.com`
   - SSL: `Let's Encrypt` (automático)
   - Force HTTPS: `Sí`

4.2 Configurar DNS

```
# En tu proveedor de DNS
Tipo: CNAME
Nombre: escalafin
Valor: tu-app.easypanel.app
TTL: 300
```

4.3 Actualizar Variables de Entorno

```
# Cambiar después de configurar el dominio
NEXTAUTH_URL=https://escalafin.tudominio.com
```

📊 Paso 5: Configuraciones de Producción

5.1 Recursos y Límites

```
# En EasyPanel → Resources
CPU Limit: 1 vCore
Memory Limit: 1GB
Storage: 5GB
Scaling: Manual

# Auto-scaling (opcional)
Min Replicas: 1
Max Replicas: 3
CPU Threshold: 80%
Memory Threshold: 85%
```

5.2 Health Checks Avanzados

```
# En EasyPanel → Health Checks
Liveness Probe:
  Path: /api/health
  Port: 3000
  Initial Delay: 30s
  Period: 30s
  Timeout: 10s
  Failure Threshold: 3

Readiness Probe:
  Path: /api/health
  Port: 3000
  Initial Delay: 10s
  Period: 10s
  Timeout: 5s
  Failure Threshold: 3
```

5.3 Logs y Monitoreo

```
# Configuración de logs
Log Level: INFO
Log Retention: 7 days
Max Log Size: 100MB

# Métricas
CPU Usage: ✅ Monitor
Memory Usage: ✅ Monitor
Disk Usage: ✅ Monitor
Response Time: ✅ Monitor
Error Rate: ✅ Monitor
```

🔒 Paso 6: Seguridad en Producción

6.1 Variables de Entorno Seguras

```
# Generar secrets seguros
NEXTAUTH_SECRET=$(openssl rand -base64 32)
DB_PASSWORD=$(openssl rand -base64 24)
```

6.2 Configurar Firewall

```
# En EasyPanel → Security
Allowed Ports:
  - 3000 (HTTP)
  - 443 (HTTPS)

Blocked IPs:
  - (configurar según necesidad)

Rate Limiting:
  - 100 requests/15min per IP
```

6.3 Backups Automáticos

```
# Base de datos
Schedule: Daily at 2:00 AM UTC
Retention: 7 days
Location: S3 bucket

# Application data
Files: Stored in S3 (already backed up)
Config: Version controlled in Git
```

🐛 Paso 7: Troubleshooting

7.1 Errores Comunes de Build

```
# Error: Cannot find module 'xyz'
# Solución: Verificar que yarn.lock está actualizado
rm -rf node_modules package-lock.json
yarn install

# Error: Prisma client not found
# Solución: Verificar que se ejecuta "npx prisma generate" en build
# (Ya incluido en Dockerfile)

# Error: Out of memory during build
# Solución: Aumentar recursos de build en EasyPanel
Build Memory: 2GB (temporal para build)
Runtime Memory: 1GB
```

7.2 Errores de Runtime

```
# Error: Database connection refused
# Verificar:
1. DATABASE_URL correcta
2. PostgreSQL service corriendo
3. Network connectivity entre services

# Error: 502 Bad Gateway
# Verificar:
1. App corriendo en puerto 3000
2. Health check respondiendo
3. No errores en application logs

# Error: NextAuth configuration
# Verificar:
1. NEXTAUTH_URL coincide con dominio real
2. NEXTAUTH_SECRET tiene al menos 32 caracteres
3. Cookies domain configuration
```

7.3 Performance Issues

```
# App lenta
# Verificar:
1. CPU/Memory usage en dashboard
2. Database query performance
3. Network latency

# High memory usage
# Verificar:
1. Memory leaks en código
2. Large file uploads
3. Cache configuration

# Database queries lentas
# Optimizar:
1. Agregar índices necesarios
2. Optimizar Prisma queries
3. Implementar caching
```

🔧 Paso 8: Desarrollo Local con Docker

8.1 Ejecutar Localmente

```
# Clonar repositorio
git clone https://github.com/tu-usuario/escalafin-mvp.git
cd escalafin-mvp

# Ejecutar con Docker Compose
docker-compose up -d

# Verificar servicios
docker-compose ps

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

8.2 Comandos Útiles

```
# Rebuild solo la app
docker-compose up --build app

# Ejecutar migraciones
docker-compose exec app npx prisma db push

# Acceder al container
docker-compose exec app sh

# Backup de DB local
docker-compose exec db pg_dump -U escalafin escalafin > backup.sql

# Restaurar backup
docker-compose exec -T db psql -U escalafin escalafin < backup.sql
```

✅ Checklist de Verificación

🏗️ Build y Deploy

- [ ] Dockerfile optimizado creado
- [ ] .dockerignore configurado correctamente
- [ ] Build completa sin errores
- [ ] Container se inicia correctamente
- [ ] Health check responde `/api/health`

🗄️ Base de Datos

- [ ] PostgreSQL service creado
- [ ] DATABASE_URL configurada
- [ ] Migraciones ejecutadas
- [ ] Seed data cargada
- [ ] Backups configurados

🌐 Networking y SSL

- [ ] App accesible por HTTP/HTTPS
- [ ] Dominio personalizado configurado
- [ ] SSL certificate válido
- [ ] Redirects HTTP→HTTPS funcionando

🔐 Seguridad

- [ ] Variables de entorno de producción
- [ ] Secrets rotados desde desarrollo
- [ ] Headers de seguridad configurados
- [ ] Rate limiting activado

📊 Monitoreo

- [ ] Health checks funcionando
- [ ] Logs visible en dashboard
- [ ] Métricas de performance activas
- [ ] Alertas configuradas

🧪 Funcionalidades

- [ ] Login/logout funciona
- [ ] CRUD de clientes funciona
- [ ] Creación de préstamos funciona
- [ ] Pagos con Openpay funcionan
- [ ] WhatsApp notifications funcionan
- [ ] Upload de archivos funciona
- [ ] Reportes se generan

🎯 Comandos Rápidos de Verificación

```
# Verificar que la app está corriendo
curl -s https://escalafin.tudominio.com/api/health | jq

# Verificar SSL
curl -I https://escalafin.tudominio.com

# Verificar logs de container
docker logs escalafin-app --tail 50

# Verificar recursos del container
docker stats escalafin-app

# Conectar a la base de datos
docker exec -it escalafin-db psql -U escalafin_user -d escalafin
```

🚀 ¡EscalaFin Desplegado con Docker en EasyPanel!

📊 Información del Despliegue

```
🐳 Container: node:18-alpine optimizado
🌐 URL: https://escalafin.tudominio.com
🗄️ Database: PostgreSQL 14 en EasyPanel
🔐 Auth: NextAuth con JWT sessions
💳 Payments: Openpay Production API
📱 WhatsApp: EvolutionAPI
🗂️ Files: AWS S3 bucket
📊 Monitoring: Health checks + Logs
🔒 Security: HTTPS + Headers + Rate limiting
```

🎉 Ventajas del Despliegue Docker

- ✅ **Consistencia**: Mismo entorno en desarrollo/producción
- ✅ **Escalabilidad**: Fácil scaling horizontal
- ✅ **Rollbacks**: Deploy/rollback rápidos
- ✅ **Aislamiento**: Dependencies contenidas
- ✅ **Performance**: Imagen optimizada multi-stage
- ✅ **Monitoreo**: Health checks integrados
- ✅ **Maintenance**: Updates sin downtime

📞 Siguientes Pasos

1. **Configurar CI/CD pipeline** para deploys automáticos
2. **Implementar logging estructurado** con niveles
3. **Configurar métricas avanzadas** con Prometheus
4. **Configurar alertas** para errores críticos
5. **Planificar estrategia de scaling** según demanda

¡Tu sistema EscalaFin está ahora en producción con Docker, optimizado y listo para escalar! 🚀💪
