



# Estrategia Completa de Deploy en EasyPanel

**Proyecto:** EscalaFin MVP

**Plataforma:** EasyPanel (Docker)

**Fecha:** 16 de octubre de 2025

**Versión:** 1.0



## Tabla de Contenidos

1. [Pre-Deployment Checklist](#)
2. [Archivos Críticos Requeridos](#)
3. [Verificación de Dependencias](#)
4. [Estrategia de Errores Comunes](#)
5. [Variables de Entorno Requeridas](#)
6. [Plan de Rollback](#)
7. [Monitoreo Post-Deploy](#)
8. [Scripts Automatizados](#)



## Pre-Deployment Checklist



### Fase 1: Verificación Local

```
# Ejecutar este script antes de hacer deploy
cd /home/ubuntu/escalafin_mvp
./scripts/pre-deploy-check.sh
```

### Checklist Manual:

- [ ] **Dockerfile existe y es válido**
  - Ruta: `/home/ubuntu/escalafin_mvp/Dockerfile`
  - Versión actual: v16.0
  - Verifica: `cat Dockerfile | head -20`
- [ ] **package.json y package-lock.json existen**
  - Ruta: `/home/ubuntu/escalafin_mvp/app/package.json`
  - Ruta: `/home/ubuntu/escalafin_mvp/app/package-lock.json`
  - lockfileVersion: 3
- [ ] **Prisma schema está completo**
  - Ruta: `/home/ubuntu/escalafin_mvp/app/prisma/schema.prisma`
  - Verifica modelos: User, Client, Loan, etc.

- [ ] **Scripts de inicio existen**
  - `start.sh` - Script de inicialización
  - `healthcheck.sh` - Script de health check
  - Ambos deben tener permisos de ejecución (`chmod +x`)
- [ ] **next.config.js configurado correctamente**
  - output: 'standalone' debe estar configurado
  - Verifica: `grep -A5 "output:" app/next.config.js`
- [ ] **Git está actualizado**
  - Sin cambios sin commitear
  - Branch: main
  - Último commit pusheado

## Archivos Críticos Requeridos

### 1. Dockerfile (OBLIGATORIO)

```
/home/ubuntu/escalafin_mvp/Dockerfile
```

**Propósito:** Define cómo construir la imagen Docker

**Versión actual:** v16.0

**Crítico:** ★★★★★

**Verificación:**

```
test -f /home/ubuntu/escalafin_mvp/Dockerfile && echo "✅ OK" || echo "❌ FALTA"
```

### 2. .dockerignore (RECOMENDADO)

```
/home/ubuntu/escalafin_mvp/.dockerignore
```

**Propósito:** Excluir archivos innecesarios del build

**Crítico:** ★★★★★

**Contenido recomendado:**

```
node_modules
.git
.env*
!.env.example
*.log
.next
dist
coverage
.vscode
.idea
*.md
!README.md
docs
test-scripts
testing-documentation
instances
escalafin-*.tar.gz
*.bundle
*_BACKUP_*
*_temp
```

---

### 3. package.json (OBLIGATORIO)

```
/home/ubuntu/escalafin_mvp/app/package.json
```

**Propósito:** Define dependencias y scripts

**Crítico:** ★★★★★

**Scripts requeridos:**

```
{
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  }
}
```

---

### 4. package-lock.json (OBLIGATORIO)

```
/home/ubuntu/escalafin_mvp/app/package-lock.json
```

**Propósito:** Lockfile de dependencias (lockfileVersion 3)

**Crítico:** ★★★★★

---

## 5. next.config.js (OBLIGATORIO)

```
/home/ubuntu/escalafin_mvp/app/next.config.js
```

**Propósito:** Configuración de Next.js

**Crítico:** ★★★★★

**Configuración requerida:**

```
module.exports = {  
  output: 'standalone', // ⚠ CRÍTICO para Docker  
  // ... otras configuraciones  
}
```

---

## 6. prisma/schema.prisma (OBLIGATORIO)

```
/home/ubuntu/escalafin_mvp/app/prisma/schema.prisma
```

**Propósito:** Schema de base de datos

**Crítico:** ★★★★★

---

## 7. start.sh (RECOMENDADO)

```
/home/ubuntu/escalafin_mvp/start.sh
```

**Propósito:** Script de inicialización con health checks

**Crítico:** ★★★★★

**Debe ser ejecutable:**

```
chmod +x /home/ubuntu/escalafin_mvp/start.sh
```

---

## 8. healthcheck.sh (RECOMENDADO)

```
/home/ubuntu/escalafin_mvp/healthcheck.sh
```

**Propósito:** Verificar salud del container

**Crítico:** ★★★

---

## 9. .env.example (RECOMENDADO)

```
/home/ubuntu/escalafin_mvp/.env.example
```

**Propósito:** Template de variables de entorno

**Crítico:** ★★★★★

---

## Verificación de Dependencias

### Script de Verificación Automática

```
#!/bin/bash
# Verificar todas las dependencias críticas

echo "🔍 Verificando dependencias..."

# 1. Node.js y npm
node --version || echo "❌ Node.js no encontrado"
npm --version || echo "❌ npm no encontrado"

# 2. Archivos críticos
FILES=(
  "Dockerfile"
  "app/package.json"
  "app/package-lock.json"
  "app/next.config.js"
  "app/prisma/schema.prisma"
  "start.sh"
  "healthcheck.sh"
)

for file in "${FILES[@]}; do
  if [ -f "$file" ]; then
    echo "✅ $file"
  else
    echo "❌ $file FALTA"
  fi
done

# 3. Verificar package.json scripts
if grep -q '"build"' app/package.json; then
  echo "✅ Script 'build' encontrado"
else
  echo "❌ Script 'build' NO encontrado"
fi

# 4. Verificar output standalone en next.config.js
if grep -q "output.*standalone" app/next.config.js; then
  echo "✅ output: 'standalone' configurado"
else
  echo "⚠️ output: 'standalone' NO configurado"
fi

# 5. Verificar lockfileVersion
LOCKFILE_VERSION=$(grep -o '"lockfileVersion": [0-9]*' app/package-lock.json | grep -o '[0-9]*')
echo "📄 lockfileVersion: $LOCKFILE_VERSION"
```

## Estrategia de Errores Comunes

### Error 1: “npm ci: lockfileVersion not supported”

Síntoma:

```
npm error The npm ci command can only install with an existing package-lock.json
```

**Causa:**

- lockfileVersion incompatible con la versión de npm

**Solución:**

```
# El Dockerfile v16.0 ya resuelve esto usando npm install
# Si persiste, verifica que estés usando la versión correcta del Dockerfile

# Verificar versión del Dockerfile
head -5 /home/ubuntu/escalafin_mvp/Dockerfile | grep "Versión:"
# Debe mostrar: Versión: 16.0 o superior
```

**Prevención:**

- ☒ Usa Dockerfile v16.0 o superior
- ☒ No uses npm ci directamente, usa npm install

**Error 2: “Cannot find module ‘next/server’”****Síntoma:**

```
Error: Cannot find module 'next/server'
```

**Causa:**

- Dependencias no instaladas correctamente
- node\_modules corrupto

**Solución:**

```
cd /home/ubuntu/escalafin_mvp/app
rm -rf node_modules package-lock.json
npm install
```

**Prevención:**

- ☒ Asegúrate de que node\_modules esté en .dockerignore
- ☒ Deja que Docker instale las dependencias desde cero

**Error 3: “Prisma Client not generated”****Síntoma:**

```
Error: @prisma/client did not initialize yet
```

**Causa:**

- Prisma generate no se ejecutó durante el build

**Solución:**

```
# Verifica que el Dockerfile incluya:  
RUN npx prisma generate
```

**Prevención:**

- ☒ El Dockerfile v16.0 ya incluye prisma generate
- ☒ Verifica en los logs del build que se ejecute

---

**Error 4: “standalone build not found”****Síntoma:**

```
Error: Cannot find standalone build
```

**Causa:**

- `output: 'standalone'` no configurado en `next.config.js`

**Solución:**

```
# Editar app/next.config.js y agregar:  
module.exports = {  
  output: 'standalone',  
  // ... resto de la configuración  
}
```

**Prevención:**

- ☒ Verifica `next.config.js` antes de cada deploy
- ☒ Usa el script de verificación pre-deploy

---

**Error 5: “Database connection failed”****Síntoma:**

```
Error: Can't reach database server
```

**Causa:**

- `DATABASE_URL` incorrecto o no configurado
- Base de datos no accesible desde el container

**Solución:**



```
# 1. Verifica DATABASE_URL en EasyPanel
# 2. Asegúrate de que la base de datos esté corriendo
# 3. Verifica que el host sea accesible desde el container

# Para EasyPanel, usa el host interno si la DB está en EasyPanel:
postgresql://user:pass@postgres:5432/dbname

# Si la DB está externa, usa la IP/host externo:
postgresql://user:pass@external-db.com:5432/dbname
```

**Prevención:**

- ☒ Prueba la conexión a la DB antes del deploy
- ☒ Usa nombres de host internos si la DB está en EasyPanel

---

**Error 6: “AWS credentials not found”****Síntoma:**

Error: Missing credentials **in** config

**Causa:**

- Variables de entorno de AWS no configuradas

**Solución:**

```
# Configurar en EasyPanel:
AWS_ACCESS_KEY_ID=tu_access_key
AWS_SECRET_ACCESS_KEY=tu_secret_key
AWS_REGION=us-east-1
AWS_BUCKET_NAME=tu_bucket
AWS_FOLDER_PREFIX=escalafin/
```

**Prevención:**

- ☒ Usa .env.example como referencia
- ☒ Verifica que todas las variables estén en EasyPanel

---

**Error 7: “NEXTAUTH\_SECRET not set”****Síntoma:**

Error: Please provide NEXTAUTH\_SECRET

**Causa:**

- NEXTAUTH\_SECRET no configurado

**Solución:**

```
# Generar un secret aleatorio:
openssl rand -base64 32

# O usar Node.js:
node -e "console.log(require('crypto').randomBytes(32).toString('base64'))"

# Configurar en EasyPanel:
NEXTAUTH_SECRET=tu_secret_generado
```

**Prevención:**

- ☒ Genera secrets únicos para cada ambiente
- ☒ NUNCA reutilices secrets entre dev y production

**Error 8: “Port already in use”****Síntoma:**

Error: Port 3000 **is** already **in** use

**Causa:**

- Puerto configurado incorrectamente en EasyPanel

**Solución:**

```
# En EasyPanel:
# 1. Ve a Settings > Ports
# 2. Configura el puerto del container: 3000
# 3. Configura el puerto público: 80 o 443
```

**Prevención:**

- ☒ Usa siempre el puerto 3000 dentro del container
- ☒ Deja que EasyPanel maneje el mapeo de puertos

**Variables de Entorno Requeridas****Variables OBLIGATORIAS**

```
# Base de Datos (CRÍTICO ★★★★★★)
DATABASE_URL=postgresql://user:password@host:5432/dbname

# NextAuth (CRÍTICO ★★★★★★)
NEXTAUTH_URL=https://tu-dominio.com
NEXTAUTH_SECRET=tu_secret_super_seguro_aqui

# Node Environment (CRÍTICO ★★★★★★)
NODE_ENV=production
```

## Variables RECOMENDADAS

```
# AWS S3 (para file uploads)
AWS_ACCESS_KEY_ID=tu_access_key
AWS_SECRET_ACCESS_KEY=tu_secret_key
AWS_REGION=us-east-1
AWS_BUCKET_NAME=tu_bucket_name
AWS_FOLDER_PREFIX=escalafin/

# Openpay (para pagos)
OPENPAY_MERCHANT_ID=tu_merchant_id
OPENPAY_PRIVATE_KEY=tu_private_key
OPENPAY_PUBLIC_KEY=tu_public_key
OPENPAY_API_ENDPOINT=https://api.openpay.mx/v1
OPENPAY_IS_PRODUCTION=true

# Evolution API (para WhatsApp)
EVOLUTION_API_URL=https://tu-evolution-api.com
EVOLUTION_API_KEY=tu_api_key
EVOLUTION_INSTANCE_NAME=tu_instancia

# Configuración Adicional
NEXT_PUBLIC_APP_URL=https://tu-dominio.com
SKIP_ENV_VALIDATION=false
```

## Cómo Configurar en EasyPanel

1. Navega a tu aplicación en EasyPanel
2. Ve a Settings > Environment
3. Agrega cada variable una por una
4. Guarda y reinicia la aplicación



## Plan de Rollback

### Estrategia de Rollback en 3 Niveles

#### Nivel 1: Rollback Rápido (< 5 minutos)

**Cuándo usar:** El deploy falló pero la versión anterior está disponible

```
# En EasyPanel:
# 1. Ve a Deployments
# 2. Encuentra el último deployment exitoso
# 3. Click en "Redeploy"
# 4. Espera a que termine
```

#### Nivel 2: Rollback por Git (< 15 minutos)

**Cuándo usar:** Necesitas revertir cambios en el código

```
# 1. Identifica el commit anterior estable
git log --oneline -10

# 2. Revierte al commit estable
git revert <commit-hash>

# 0 resetea (si no has pusheado):
git reset --hard <commit-hash>

# 3. Push
git push origin main

# 4. Redeploy en EasyPanel
```

### Nivel 3: Rollback Completo (< 30 minutos)

**Cuándo usar:** Todo falló y necesitas restaurar desde backup

```
# 1. Restaurar código desde backup
cp -r /home/ubuntu/escalafin_mvp_BACKUP_* /home/ubuntu/escalafin_mvp

# 2. Restaurar base de datos
# Ver script: restore-db.sh

# 3. Verificar y redeploy
cd /home/ubuntu/escalafin_mvp
./scripts/pre-deploy-check.sh
# Redeploy en EasyPanel
```



## Monitoreo Post-Deploy

### Checklist de Verificación Post-Deploy

**Inmediatamente después del deploy (0-5 minutos):**

- ☐ **Container está corriendo**
  - En EasyPanel: Status = Running (verde)
- ☐ **Logs no muestran errores críticos**
  - En EasyPanel: Ve a Logs
  - Busca: “Server started”, “Listening on port 3000”
  - No debe haber: “Error”, “FATAL”, “cannot connect”
- ☐ **Health check pasa**
  - En EasyPanel: Ve a Health Checks
  - Status debe ser: Healthy
- ☐ **URL responde**
  - Abre: <https://tu-dominio.com>
  - Debe cargar la página de login

**30 minutos después del deploy:**

- ☐ **Login funciona**
- Prueba login con usuario de prueba
- ☐ **Database queries funcionan**
- Verifica que el dashboard cargue datos
- ☐ **File uploads funcionan** (si aplica)
- Sube un documento de prueba
- ☐ **Payments funcionan** (si aplica)
- Crea una transacción de prueba

**24 horas después del deploy:**

- ☐ **No memory leaks**
- En EasyPanel: Ve a Metrics
- Memoria debe ser estable, no creciendo constantemente
- ☐ **No errores recurrentes en logs**
- Revisa logs para patrones de error
- ☐ **Performance acceptable**
- Tiempos de respuesta < 2 segundos

## Scripts Automatizados

**Script 1: Pre-Deploy Check**

Ver: `/home/ubuntu/escalafin_mvp/scripts/pre-deploy-check.sh`

**Script 2: Post-Deploy Verification**

Ver: `/home/ubuntu/escalafin_mvp/scripts/post-deploy-check.sh`

**Script 3: Emergency Rollback**

Ver: `/home/ubuntu/escalafin_mvp/scripts/emergency-rollback.sh`

## Soporte y Recursos

**Documentación de Referencia**

1. **MULTI\_INSTANCE\_GUIDE.md** - Guía de deployment multi-instancia
2. **COOLIFY\_DEPLOYMENT\_GUIDE.md** - Guía de Coolify (similar a EasyPanel)
3. **EASYPANEL\_DOCKER\_GUIDE.md** - Guía específica de EasyPanel
4. **FIX\_NPM\_CI\_LOCKFILEVERSION.md** - Fix del error npm ci

## Logs para Debugging

```
# En EasyPanel, accede a:
# 1. Logs en tiempo real
# 2. Logs históricos (últimas 24 horas)
# 3. Métricas de performance

# Filtrar logs por nivel:
# - ERROR: Solo errores críticos
# - WARN: Advertencias
# - INFO: Información general
```

## Comandos Útiles de Debugging

```
# Ver estado del container
docker ps -a | grep escalafin

# Ver logs del container
docker logs <container-id> --tail 100

# Entrar al container
docker exec -it <container-id> sh

# Verificar variables de entorno
docker exec <container-id> env | grep -E "DATABASE|NEXTAUTH|AWS"

# Verificar conectividad a DB
docker exec <container-id> nc -zv postgres-host 5432
```

## Checklist Final

### Antes de Hacer Deploy

- ☐ Código commiteado y pusheado a GitHub
- ☐ Dockerfile v16.0 o superior
- ☐ Todas las variables de entorno configuradas en EasyPanel
- ☐ Base de datos está accesible
- ☐ Backup reciente existe
- ☐ Pre-deploy check ejecutado sin errores

### Durante el Deploy

- ☐ Monitorear logs en tiempo real
- ☐ Verificar que cada stage del build completa
- ☐ No cerrar la ventana hasta que termine

### Después del Deploy











- ☐ Container en estado "Running"
- ☐ Health check pasa
- ☐ URL principal carga
- ☐ Login funciona
- ☐ Sin errores críticos en logs

---






## Resumen Ejecutivo

---

### Deploy Exitoso en 10 Pasos

1.  Ejecutar `pre-deploy-check.sh`
2.  Verificar Dockerfile v16.0
3.  Commit y push a GitHub
4.  Configurar variables de entorno en EasyPanel
5.  Iniciar deploy desde EasyPanel
6.  Monitorear logs durante build
7.  Esperar a que container esté “Running”
8.  Verificar health check
9.  Probar URL principal
10.  Ejecutar `post-deploy-check.sh`

### En Caso de Error

1.  Revisar logs en EasyPanel
2.  Consultar “Estrategia de Errores Comunes” arriba
3.  Aplicar fix correspondiente
4.  Si no funciona, ejecutar rollback
5.  Documentar el error para futuras referencias

---

**Última actualización:** 16 de octubre de 2025

**Versión:** 1.0

**Mantenido por:** Equipo EscalaFin