🐳 COMANDOS PARA TEST LOCAL CON DOCKER

Si tienes Docker instalado en tu máquina, puedes hacer un test completo del build antes de desplegar en EasyPanel.

📋 Pre-requisitos

```
# 1. Verificar que Docker esté instalado
docker --version

# 2. Verificar que Docker esté corriendo
docker ps

# 3. Clonar el repositorio (si no lo tienes)
git clone https://github.com/qhosting/escalafin-mvp.git
cd escalafin-mvp
```

🔨 Test Build Local con Docker

Opción 1: Build y Run Completo

```
# 1. Hacer build de la imagen
docker build -t escalafin-test:local .

# 2. Verificar que la imagen se creó
docker images | grep escalafin-test

# 3. Ejecutar el contenedor
docker run -d \
  --name escalafin-test \
  -p 3000:3000 \
  -e DATABASE_URL="postgresql://..." \
  -e NEXTAUTH_SECRET="test-secret-12345" \
  -e NEXTAUTH_URL="http://localhost:3000" \
  -e NODE_ENV="production" \
  escalafin-test:local

# 4. Ver logs en tiempo real
docker logs -f escalafin-test

# 5. Verificar que esté corriendo
curl http://localhost:3000/api/health

# 6. Abrir en navegador
# → http://localhost:3000
```

Opción 2: Build con Verificación Paso a Paso

```
# Build con output detallado
docker build --no-cache --progress=plain -t escalafin-test:local . 2>&1 | tee docker-build.log

# Revisar el log completo
less docker-build.log

# Buscar errores específicos
grep -i error docker-build.log
grep -i warning docker-build.log
```

🔍 Verificación del Build

1. Inspeccionar la Imagen

```
# Ver capas de la imagen
docker history escalafin-test:local

# Inspeccionar la imagen
docker inspect escalafin-test:local | grep -A 10 "Env"
```

2. Inspeccionar el Contenedor (sin ejecutarlo)

```
# Crear contenedor sin iniciarlo
docker create --name escalafin-inspect escalafin-test:local

# Ver el sistema de archivos
docker export escalafin-inspect | tar t | grep -E "(server.js|package.json|.next)"

# Limpiar
docker rm escalafin-inspect
```

3. Ejecutar Shell Dentro del Contenedor

```
# Ejecutar bash dentro del contenedor
docker run -it --rm escalafin-test:local /bin/bash

# Una vez dentro:
ls -la /app/
ls -la /app/.next/standalone/
cat /app/start.sh
exit
```

🧪 Pruebas Funcionales

Test del API

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

# Test de autenticación (debe redirigir a login)
curl -I http://localhost:3000/admin/dashboard

# Verificar que static assets cargan
curl -I http://localhost:3000/_next/static/...
```

Test de UI

1. Abrir en navegador: `http://localhost:3000`
2. Verificar que carga la página de login
3. Intentar login con credenciales de prueba
4. Verificar que las rutas funcionan

🧹 Limpieza

```
# Detener contenedor
docker stop escalafin-test

# Eliminar contenedor
docker rm escalafin-test

# Eliminar imagen
docker rmi escalafin-test:local

# Limpiar todo (imágenes huérfanas, cache, etc)
docker system prune -a --volumes
```

📊 Monitoreo

Ver Logs

```
# Logs en tiempo real
docker logs -f escalafin-test

# Últimas 100 líneas
docker logs --tail 100 escalafin-test

# Logs con timestamps
docker logs -t escalafin-test
```

Ver Recursos

```
# Uso de CPU y memoria
docker stats escalafin-test

# Procesos corriendo en el contenedor
docker top escalafin-test
```

❌ Troubleshooting

El Build Falla

```
# 1. Verificar Dockerfile
cat Dockerfile | head -20

# 2. Build sin cache para ver todos los pasos
docker build --no-cache --progress=plain -t escalafin-test:local .

# 3. Verificar que el repo está actualizado
git pull origin main
git log -1 --oneline
```

El Contenedor No Inicia

```
# Ver logs de error
docker logs escalafin-test

# Verificar si el puerto está ocupado
netstat -tuln | grep 3000
# o
lsof -i :3000

# Ejecutar con diferentes variables
docker run -it --rm \
  -e NODE_ENV=development \
  -e DEBUG=* \
  escalafin-test:local
```

La App No Responde

```
# Verificar que el contenedor está corriendo
docker ps | grep escalafin

# Verificar la red
docker inspect escalafin-test | grep -A 10 "NetworkSettings"

# Probar con exec
docker exec -it escalafin-test curl http://localhost:3000/api/health

# Ver procesos dentro del contenedor
docker exec -it escalafin-test ps aux
```

🎯 Resultado Esperado

Si todo funciona correctamente, deberías ver:

En Build Logs

```
✓ Compiled successfully
Route (app)                                 Size     First Load JS
┌ ƒ /                                       4.8 kB          119 kB
...
Successfully tagged escalafin-test:local
```

En Runtime Logs

```
▲ Next.js 14.2.28
- Local:        http://0.0.0.0:3000
✓ Ready in XXXms
```

En Navegador

- ✅ Página de login carga correctamente
- ✅ Estilos y assets se cargan
- ✅ No hay errores 404 o 500
- ✅ API responde correctamente

💡 Equivalencia con EasyPanel

El test local con Docker simula exactamente lo que hace EasyPanel:

Local	EasyPanel
`docker build`	Build automático desde GitHub
Variables `-e`	Variables de entorno en Settings
`-p 3000:3000`	Port mapping en configuración
`docker logs`	Logs en EasyPanel UI
Acceso localhost:3000	Acceso via dominio configurado

Si funciona local, funcionará en EasyPanel ✅

📝 Notas Importantes

1. **Database URL**: Usa la misma DATABASE_URL que tendrás en producción
2. **Secrets**: Usa secrets temporales para testing, no los de producción

3. **Puerto 3000**: El Dockerfile está configurado para puerto 3000
4. **Limpieza**: Recuerda limpiar imágenes viejas para no llenar disco

---

Próximo paso: Si el test local funciona, procede con el deploy en EasyPanel siguiendo `DIAGNOSTICO_RUNTIME_EASYPANEL.md`