

✓ PUSH EXITOSO - Siguiente Paso: Re-Deploy en Coolify

🎉 ESTADO ACTUAL

Push Completado Exitosamente

- ✓ Conexión SSH con GitHub verificada
- ✓ Remote cambiado de HTTPS a SSH
- ✓ 10 commits subidos correctamente a GitHub
- ✓ Repositorio actualizado

Commits Subidos

```
e8778f3 - Checkpoint automático
81176aa - Checkpoint automático
e85b810 - docs: instrucciones para autenticación y push a GitHub
0968668 - Checkpoint automático
9faa0b3 - docs: resumen ejecutivo del fix
e99fd9d - docs: mensaje final completo para fix build error
1d1ec39 - docs: agregar archivo de acción inmediata para fix build
0bd3c70 - fix: Dockerfile.coolify v11.0 (CRÍTICO) 🚨
76ce82a - Docker build fixed NPM
54bfc2e - Fix Docker yarn.lock symlink error
```

Commit Más Importante

0bd3c70 - fix: Dockerfile.coolify v11.0

Este commit contiene:

- ✓ Migración de Yarn/NPM dual a solo NPM
- ✓ Eliminación de lógica condicional problemática
- ✓ Configuración estable para Coolify
- ✓ Optimizaciones de build

🎯 SIGUIENTE PASO: RE-DEPLOY EN COOLIFY

Opción 1: Re-Deploy desde la UI de Coolify (Recomendado)

1. Abrir Coolify:

<https://adm.escalafin.com>

2. Navegar al proyecto:

- Dashboard → Projects
- Seleccionar “EscalaFin MVP” (o el nombre de tu proyecto)

3. Hacer Re-Deploy:

- Click en el botón “Redeploy” o “Deploy”

- Coolify detectará automáticamente los cambios en GitHub
- Comenzará el build con el nuevo `Dockerfile.coolify`

4. Monitorear el Build:

- Hacer click en **“Logs”** o **“Build Logs”**
- Buscar estas líneas clave:

```
✓ Instalando dependencias con NPM...
✓ NPM install completado
✓ Building Next.js application...
✓ Compiled successfully
✓ Generating static pages...
✓ Build completed
```

1. Verificar Éxito:

- Estado del deploy: **“Running”** o **“Healthy”**
- No debería haber errores de `ENOENT: no such file or directory, open 'yarn.lock'`

Opción 2: Re-Deploy Manual via SSH (Avanzado)

Si tienes acceso SSH al servidor de Coolify:

```
# Conectar al servidor
ssh usuario@adm.escalafin.com

# Navegar al directorio del proyecto (ajustar ruta según tu instalación)
cd /path/to/coolify/apps/escalafin-mvp

# Pull de los cambios
git pull origin main

# Rebuild el contenedor
docker-compose down
docker-compose up -d --build

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

VERIFICACIÓN POST-DEPLOY

Después del deploy, verifica lo siguiente:

1. Aplicación Funcionando

```
# Hacer una prueba de la URL del proyecto
curl https://tu-escalafin-url.com/api/health
```

Deberías recibir respuesta exitosa (200 OK)

2. Logs Limpios

Verificar que no haya:

-  Errores de `yarn.lock`

- ❌ Errores de `ENOENT`
- ❌ Errores de instalación de dependencias

Deberías ver:

- ✅ NPM install exitoso
- ✅ Build completado
- ✅ Aplicación iniciada

3. Funcionalidades Principales

- ☐ Login funciona
- ☐ Dashboard carga correctamente
- ☐ Base de datos conectada
- ☐ API endpoints responden
- ☐ Archivos estáticos se sirven



CONFIGURACIÓN DE COOLIFY

Variables de Entorno Requeridas

Asegúrate de que estas variables estén configuradas en Coolify:

```
# Base de datos
DATABASE_URL=postgresql://...

# NextAuth
NEXTAUTH_URL=https://tu-dominio.com
NEXTAUTH_SECRET=...

# AWS S3 (si aplica)
AWS_BUCKET_NAME=...
AWS_FOLDER_PREFIX=...

# Openpay (si aplica)
OPENPAY_API_KEY=...
OPENPAY_MERCHANT_ID=...
```

Dockerfile que Se Usará

Coolify usará automáticamente: `Dockerfile.coolify`

Este es el archivo que acabamos de corregir (v11.0) que:

- Usa solo NPM (no Yarn)
 - Tiene mejor manejo de dependencias
 - Es más estable para producción
-

TROUBLESHOOTING

Si el Build Falla

1. Verificar logs en Coolify:

- Buscar mensajes de error específicos
- Capturar la sección del error

2. Problemas comunes:

Error: `yarn.lock not found`

- ❌ Coolify está usando el Dockerfile viejo
- Solución: Forzar rebuild con `docker-compose build --no-cache`

Error: `npm install failed`

- ❌ Problema de dependencias o red
- Solución: Verificar que `package.json` esté correcto

Error: `Next.js build failed`

- ❌ Errores de TypeScript o código
- Solución: Revisar el código fuente del error

1. Verificar Dockerfile correcto:

```
```bash
```

```
En el servidor de Coolify
```

```
cat Dockerfile.coolify | head -20
```

```
Debería mostrar la versión v11.0
```

```
Con comentarios sobre NPM-only
```

```
```
```

Si la Aplicación No Inicia

1. Verificar puerto:

- Coolify debe mapear el puerto 3000 del contenedor

2. Verificar health check:

- Coolify debe poder hacer ping a `/api/health` o `/`

3. Verificar variables de entorno:

- Todas las variables críticas deben estar configuradas

MONITOREO POST-DEPLOY

Durante las Primeras 24 Horas





- Monitorear logs de errores
- Verificar performance de la aplicación
- Revisar métricas de uso de CPU/memoria
- Validar que todos los módulos funcionan

Métricas Clave

- **Tiempo de build:** Debería ser ~5-10 minutos

- **Tiempo de inicio:** Debería ser ~30 segundos
- **Uso de memoria:** ~512MB-1GB (depende del tráfico)
- **CPU:** < 50% en idle

RESUMEN DE ACCIÓN INMEDIATA

1.  **Hecho:** Push exitoso a GitHub
2.  **Ahora:** Ir a Coolify y hacer **Re-Deploy**
3.  **Después:** Monitorear logs del build
4.  **Finalmente:** Verificar que la aplicación funcione

ENLACES ÚTILES

- **Coolify Dashboard:** <https://adm.escalafin.com>
- **Repositorio GitHub:** <https://github.com/qhosting/escalafin-mvp>
- **Commits recientes:** <https://github.com/qhosting/escalafin-mvp/commits/main>
- **Documentación Coolify:** <https://coolify.io/docs>

NOTAS ADICIONALES

Para Futuros Pushes

Ahora que SSH está configurado, los futuros pushes son simples:

```
cd /home/ubuntu/escalafin_mvp
git add .
git commit -m "tu mensaje"
git push origin main
```

No se necesitará autenticación, todo está configurado.

Para Futuros Deploys

Coolify puede configurarse para **auto-deploy** en cada push:

1. En Coolify UI
2. Project Settings
3. Habilitar "Auto Deploy on Git Push"
4. Ahora cada push a `main` disparará un deploy automático

Estado:  Push completado - Listo para Re-Deploy

Siguiente: Ir a Coolify y hacer **Redeploy**

Tiempo estimado: 5-10 minutos para build completo
