

Diagnóstico y Solución - Error de Build en EasyPanel

Fecha: 27 de octubre de 2025

Problema: Build fallando en EasyPanel sin mostrar el error real

Estado:  RESUELTO

Problema Identificado

Error Original

```
ERROR: process "/bin/bash -c echo \"🔧 Building Next.js...\..." did not complete successfully: exit code: 1
```

El error ocurría durante el paso de build de Next.js, pero el log no mostraba el error real debido a que el output de Docker estaba siendo truncado.

Causa Raíz Encontrada

Al analizar el código, encontramos **DOS PROBLEMAS**:

1. `dynamic` export en posición incorrecta (app/app/layout.tsx)

```
//  INCORRECTO - dynamic export en medio de imports
import type { Metadata } from 'next'
import { Inter } from 'next/font/google'
import './globals.css'

export const dynamic = 'force-dynamic'; //  MAL UBICADO
import { Providers } from './providers' // Imports después del export
```

Problema: Next.js requiere que los exports de configuración de ruta (`dynamic` , `revalidate` , etc.) estén **después de todos los imports**, no en medio.

2. Dockerfile con logging complejo

El Dockerfile tenía un script complejo para capturar errores que estaba siendo truncado por Docker, impidiendo ver el error real.

✓ Solución Aplicada

1. Corregir posición del `dynamic export`

```
// ✓ CORRECTO - dynamic export después de todos los imports
import type { Metadata } from 'next'
import { Inter } from 'next/font/google'
import './globals.css'
import { Providers } from './providers'
import { Toaster } from 'sonner'
import { MainLayout } from '@components/layout/main-layout'

export const dynamic = 'force-dynamic'; // ✓ BIEN UBICADO

const inter = Inter({ subsets: ['latin'] })
```

2. Simplificar comando de build en Dockerfile

```
# ✓ NUEVO - Simple y directo
RUN echo "🔧 Building Next.js..." && \
  echo "Node version: $(node --version)" && \
  echo "Yarn version: $(yarn --version)" && \
  echo "NODE_ENV: $NODE_ENV" && \
  echo "Working directory: $(pwd)" && \
  echo "" && \
  yarn build && \
  echo "✓ Build completado"
```

Ventajas:

- Muestra errores directamente sin procesamiento complejo
- No hay truncamiento de logs
- Más fácil de depurar

📝 Commits Aplicados

Commit 1: Corregir estructura del código

```
commit d7a539c
fix: Corregir posición de dynamic export y simplificar Dockerfile

- Mover export const dynamic después de todos los imports en layout.tsx
- Simplificar comando de build en Dockerfile para mejor visibilidad de errores
- El dynamic export en medio de imports causaba error de build en Next.js
```

Commit 2: Convertir yarn.lock

```
commit 422a2c0
fix: Convertir yarn.lock a archivo regular

- Pre-push hook detectó y corrigió automáticamente el symlink
- Esencial para que Docker pueda copiar el archivo
```

Pasos para Rebuild en EasyPanel

1. Limpiar Cache de Build

- Ve a tu proyecto en EasyPanel
- Busca la opción “Clear Build Cache” o similar
- Esto asegura que use el código nuevo

2. Verificar que esté usando el commit correcto

- Último commit: `422a2c0`
- Branch: `main`
- Verifica que EasyPanel esté apuntando a este commit


3. Rebuild del Proyecto

- Haz clic en “Rebuild” o “Deploy”
- Monitorea los logs

4. Verificar el Build Log

Deberías ver ahora:

```

 Building Next.js...

Node version: v20.x.x

Yarn version: x.x.x

NODE\_ENV: production

Working directory: /app

[next build output...]


 Build completado

```



Qué Buscar en los Logs

Señales de Éxito

-  Build completado
- `Compiled successfully`
- Creación del directorio `.next/standalone`
- Sin errores de TypeScript o ESLint

Si Aún Hay Errores

Si ves errores después de estos cambios, ahora serán **visibles y claros**:

- Errores de TypeScript aparecerán directamente
 - Errores de compilación de Next.js se mostrarán completos
 - Problemas de dependencias serán evidentes
-



Estado del Proyecto

Archivos Modificados

- ☒ Dockerfile - Simplificado para mejor debugging
- ☒ app/app/layout.tsx - Corregida posición de `dynamic` export
- ☒ app/yarn.lock - Convertido de symlink a archivo regular

GitHub

- ☒ Todos los cambios pushed a `main`
- ☒ Commit hash: `422a2c0`
- ☒ Pre-push hooks funcionando correctamente

Configuración EasyPanel Necesaria

```
Build Method: Dockerfile
Build Path: /
Dockerfile Path: Dockerfile
Context Path: .
Memory: 2GB (mínimo recomendado)
```



Próximos Pasos

1. Inmediato:

- Limpiar cache en EasyPanel
- Rebuild del proyecto
- Verificar que el build complete exitosamente

2. Después del Build Exitoso:

- Verificar que la aplicación inicie correctamente
- Comprobar health check en `/api/health`
- Probar login con credenciales de prueba

3. Validación Final:

- Verificar que todas las rutas funcionen
- Comprobar que S3 local storage esté operando
- Validar que la base de datos PostgreSQL se conecte



Documentación Relacionada

- `FIX_NODE_MODULES_NOT_FOUND.md` - Fix anterior del runtime
 - `ESTADO_ACTUAL_RESUELTO.md` - Estado del proyecto
 - `DEBUGGING_BUILD_FAILURE.md` - Guía general de debugging
 - `CONFIGURACION_EASYPANEL_CORRECTA.md` - Configuración de EasyPanel
-



Lecciones Aprendidas

1. Orden de Exports en Next.js:

- Los exports de configuración de ruta deben ir después de todos los imports
- No mezclar imports con exports de configuración

2. Dockerfile Debugging:

- Menos es más: comandos simples muestran errores mejor
- Docker puede truncar logs complejos

3. Pre-push Hooks:

- Los hooks automáticos previenen problemas comunes
- Siempre verificar que yarn.lock sea un archivo regular

✓ Con estos cambios, el build debería completarse exitosamente en EasyPanel.

🔄 Última actualización: 27 de octubre de 2025, 22:15 UTC