

# FIX: Captura Correcta del Exit Code del Build

## Problema Identificado

El Dockerfile no estaba capturando correctamente el código de salida del build de Next.js debido al uso de `| tee build.log con ||`.

### El Comando Problemático

```
RUN npm run build 2>&1 | tee build.log && \
    echo " Build completed successfully!" || \
    (echo " Build failed!" && exit 1)

RUN if [ -f ".next/BUILD_ID" ]; then ...
```

**Problemas:** 1. `tee` puede retornar exit code 0 incluso si `npm run build` falla. 2. El operador `||` no se ejecutaba porque `tee` siempre tenía éxito 3. La verificación del `BUILD_ID` estaba en un `RUN` separado, que se ejecutaba incluso si el build fallaba

---

## Solución Implementada

### Usar PIPESTATUS para Capturar el Exit Code Real

```
RUN echo " Building Next.js application..." && \
    npm run build 2>&1 | tee build.log; \
    BUILD_EXIT_CODE=${PIPESTATUS[0]}; \
    if [ $BUILD_EXIT_CODE -ne 0 ]; then \
        echo " Build failed with exit code $BUILD_EXIT_CODE!"; \
        echo "Last 50 lines of build log:"; \
        tail -50 build.log; \
        exit 1; \
    fi && \
    echo " Build completed successfully!" && \
    if [ -f ".next/BUILD_ID" ]; then \
        echo " Build ID found: $(cat .next/BUILD_ID)"; \
    else \
        echo " BUILD_ID not found!"; \
        echo "Contents of .next directory:"; \
        ls -la .next/; \
        exit 1; \
    fi
```

**Mejoras:** 1. Usa  `${PIPESTATUS[0]}`  para capturar el exit code de `npm run build` 2. Verifica explícitamente si el build falló antes de continuar 3.

Consolida build y verificación en un solo RUN command 4. Muestra contenido de .next/ si BUILD\_ID no existe

---

## Cómo Funciona

### PIPESTATUS en Bash

PIPESTATUS es un array que contiene los códigos de salida de todos los comandos en un pipeline:

```
npm run build 2>&1 | tee build.log
# PIPESTATUS[0] = exit code de npm run build
# PIPESTATUS[1] = exit code de tee
```

Capturamos inmediatamente PIPESTATUS[0] porque se sobrescribe con cada nuevo pipeline.

### Flujo de Ejecución

1. Ejecutar build: `npm run build 2>&1 | tee build.log`
  2. Capturar exit code: `BUILD_EXIT_CODE=${PIPESTATUS[0]}`
  3. Verificar si falló:
    - Si `BUILD_EXIT_CODE != 0` → Mostrar últimas 50 líneas y salir con error
    - Si `BUILD_EXIT_CODE == 0` → Continuar
  4. Verificar BUILD\_ID:
    - Si existe .next/BUILD\_ID → Mostrar ID y continuar
    - Si NO existe → Listar contenido de .next/ y salir con error
- 

## Resultado Esperado

### Build Exitoso

```
Building Next.js application (NORMAL mode, no standalone)...
Next.js 14.2.28
Creating an optimized production build ...
Compiled successfully
Generating static pages (26/26)
Build completed successfully!
Build ID found: abc123xyz
```

### Build Fallido

```
Building Next.js application (NORMAL mode, no standalone)...
Error: Module not found...
```

```
Build failed with exit code 1!
Last 50 lines of build log:
[últimas 50 líneas del error]
```

## BUILD\_ID No Encontrado

```
Build completed successfully!
BUILD_ID not found!
Contents of .next directory:
total 12
drwxr-xr-x 3 root root 4096 Oct 11 18:00 .
drwxr-xr-x 8 root root 4096 Oct 11 18:00 ..
drwxr-xr-x 2 root root 4096 Oct 11 18:00 cache
```

---

## Historial de Fixes

Versión	Problema	Solución
v1	BUILD_ID en .build/	ENV NEXT_DIST_DIR=".next"
v2	String vacío rechazado	No establecer NEXT_OUTPUT_MODE
v3	Exit code no capturado	PIPESTATUS[0] + consolidar RUN

---

## Verificación

Después del redeploy, el build debería:

1. Mostrar todo el output del build via `tee`
2. Detectar si el build falla y detenerse
3. Verificar que `BUILD_ID` existe antes de continuar
4. Mostrar diagnóstico claro si algo falla

---

## Próximos Pasos

1. Push a GitHub
2. Redeploy en Coolify
3. Verificar logs del build
4. Confirmar que el container inicia correctamente

Fecha: 2025-10-11