



Fix v9.4 - Standalone Output Forzado



Problema Identificado

El build de Next.js se completaba exitosamente, pero **no generaba el directorio** `.next/standalone` :

```
ERROR: failed to solve:
if [ ! -d ".next/standalone" ]; then
  echo "❌ ERROR: standalone output no generado"
  exit 1
fi
```



Análisis del Problema

Configuración Actual

next.config.js (línea 6):

```
output: process.env.NEXT_OUTPUT_MODE,
```

Dockerfile (línea 44):

```
ENV NEXT_OUTPUT_MODE=standalone
```

¿Por Qué Fallaba?

Aunque la variable de entorno estaba configurada en el Dockerfile, hay varios escenarios donde Next.js podría no leerla correctamente durante el build:

1. **Timing:** La variable podría no estar disponible cuando Next.js lee la configuración
2. **Contexto:** El proceso de Node.js podría no heredar correctamente las variables del shell
3. **Cache:** Next.js podría estar usando una configuración en caché

Evidencia

El build completaba sin errores, pero la verificación fallaba:

```
=== Verificando build standalone ===
ls -la .next/
# .next/ existe pero no tiene subdirectorio 'standalone'
```

Esto confirma que Next.js **no estaba usando** `output: 'standalone'` durante el build.



Solución Implementada

Enfoque: Forzar Standalone en el Dockerfile

En lugar de depender de que la variable de entorno se lea correctamente, **modificamos directamente el** `next.config.js` **durante el build** del Docker:

```
# Forzar configuración standalone en next.config.js
RUN echo "=== Configurando standalone output ===" && \
  sed -i "s/output: process.env.NEXT_OUTPUT_MODE,/output: 'standalone',/" next.config.js && \
  echo "Configuración aplicada:" && \
  grep "output:" next.config.js
```

¿Qué Hace Este Comando?

1. **sed -i** : Edita el archivo in-place (directamente)
2. **"s/X/Y/"** : Reemplaza X por Y
3. **Busca:** `output: process.env.NEXT_OUTPUT_MODE,`
4. **Reemplaza con:** `output: 'standalone',`
5. **Verifica:** Imprime la línea modificada para confirmar

Resultado

Antes del sed:

```
output: process.env.NEXT_OUTPUT_MODE,
```

Después del sed:

```
output: 'standalone',
```

Por Qué Esto Funciona

- **✓ Garantizado:** El valor está hardcoded en el archivo
- **✓ Timing:** Se aplica justo antes del build
- **✓ Sin dependencias:** No depende de variables de entorno
- **✓ Visible:** El `grep` muestra que se aplicó correctamente
- **✓ Persistente:** Se mantiene durante todo el build



Comparación

Enfoque Anterior (v9.3)

```
ENV NEXT_OUTPUT_MODE=standalone

# ... más tarde ...

RUN npm run build
# Next.js lee next.config.js
# output: process.env.NEXT_OUTPUT_MODE ← Puede ser undefined
```

Problema: Variable de entorno no se lee consistentemente

Enfoque Nuevo (v9.4)

```
RUN sed -i "s/output: process.env.NEXT_OUTPUT_MODE,/output: 'standalone',/"
next.config.js

# ... inmediatamente después ...

RUN npm run build
# Next.js lee next.config.js
# output: 'standalone' ← Garantizado
```

Solución: Valor forzado directamente en el código

Flujo Completo del Build (v9.4)

```
# 1. Instalar dependencias
=== Instalando dependencias ===
npm install --legacy-peer-deps
✓ Dependencias instaladas

# 2. Generar Prisma Client
=== Generando Prisma Client ===
npx prisma generate
✓ Prisma Client generado

# 3. Forzar standalone (NUEVO)
=== Configurando standalone output ===
sed -i "s/output: process.env.NEXT_OUTPUT_MODE,/output: 'standalone',/" next.config.js
Configuración aplicada:
  output: 'standalone',
✓ Configuración standalone forzada

# 4. Build de Next.js
=== Iniciando build de Next.js ===
npm run build
✓ Compiled successfully
✓ Build completado

# 5. Verificar standalone
=== Verificando build standalone ===
ls -la .next/standalone
✓ Standalone output verificado

# 6. Copiar al stage final
✓ Aplicación lista para producción
```

Verificación del Fix

Logs Esperados en EasyPanel

Cuando ejecutes el rebuild, deberías ver:

```
Step X: Configurando standalone output
=== Configurando standalone output ===
Configuración aplicada:
  output: 'standalone',
```

Esto confirma que el `next.config.js` fue modificado correctamente.

Luego el Build Debería Completar

```
Step Y: Build de Next.js
=== Iniciando build de Next.js ===
Creating an optimized production build ...
✓ Compiled successfully
✓ Collecting page data
✓ Generating static pages
✓ Finalizing page optimization

Output Mode: standalone ← Esto confirma que funciona
```

Y Finalmente la Verificación Pasará

```
Step Z: Verificando build standalone
=== Verificando build standalone ===
drwxr-xr-x ... .next/standalone/
✓ Standalone output verificado
```



Por Qué Usar sed en Lugar de Editar el Archivo

Opción 1: Editar next.config.js Directamente

```
// next.config.js
output: 'standalone', // Hardcodeado
```

Pros:

- Simple
- Directo

Contras:

- ✗ Menos flexible (no se puede cambiar sin editar código)
- ✗ No se puede usar otro output mode en desarrollo
- ✗ Requiere modificar el archivo fuente

Opción 2: Usar ENV con Fallback

```
// next.config.js
output: process.env.NEXT_OUTPUT_MODE || 'standalone',
```

Pros:

- Flexible
- Con fallback






Contras:

- ⚠ Aún depende de que Node.js lea correctamente las ENV vars

Opción 3: sed en Dockerfile (ELEGIDA)

```
RUN sed -i "s/output: process.env.NEXT_OUTPUT_MODE,/output: 'standalone',/"
next.config.js
```

Pros:





-  Garantizado en Docker
-  Flexible en desarrollo (usa ENV)
-  No modifica el código fuente
-  Se aplica solo durante el build de Docker
-  Verificable con grep

Contras:

- Ninguno para este caso de uso



Progreso de los Fixes

Fix	Versión	Problema	Solución	Estado
#1	v9.1	NEXT_OUTPUT_MODE ENV	Agregada variable ENV	 Insuficiente
#2	v9.2	npm ci incompatible	npm install sin lock	
#3	v9.3	Sin logs de errores	Logs detallados + vars	
#4	v9.4	Standalone no se genera	sed fuerza standalone	



Impacto del Fix

Antes (v9.3)

```
Build completa → Verificación → FALLA
"standalone output no generado"
Causa: next.config.js no usa 'standalone'
```

Después (v9.4)

```
sed fuerza standalone → Build completa → Verificación → ÉXITO
Standalone output generado correctamente
```



Archivos Modificados

1. Dockerfile

Línea 3: Versión actualizada a 9.4

Líneas 68-72: Nuevo paso para forzar standalone con sed

```
# Forzar configuración standalone en next.config.js
RUN echo "=== Configurando standalone output ===" && \
  sed -i "s/output: process.env.NEXT_OUTPUT_MODE,/output: 'standalone',/" next.config.js && \
  echo "Configuración aplicada:" && \
  grep "output:" next.config.js
```

Próximos Pasos

1. Push a GitHub

```
git add Dockerfile FIX_STANDALONE_v9.4.md
git commit -m "🔧 Fix v9.4: Forzar standalone output con sed"
git push
```

2. Rebuild en EasyPanel

Esta vez el build **debería completar exitosamente** y generar el standalone.

3. Verificar los Logs

Busca estas líneas en los logs de EasyPanel:

```
=== Configurando standalone output ===
Configuración aplicada:
output: 'standalone',
```

Si ves esto, el fix se aplicó correctamente.

4. Aplicación Funcionando

Si todo sale bien:

- ☒ Build completa sin errores
- ☒ Standalone output generado
- ☒ Servidor inicia correctamente
- ☒ Aplicación accesible en tu dominio

Troubleshooting

Si el sed Falla

Error posible:

```
sed: no such file or directory
```

Causa: El archivo next.config.js no existe o está en otra ubicación

Solución: Verificar que el WORKDIR sea correcto (debería ser `/app/app`)

Si el grep No Muestra Nada

Error posible:


```
grep "output:" next.config.js
(sin output)
```

Causa: La línea no se encontró para reemplazar

Solución: El next.config.js cambió. Actualizar el patrón de búsqueda del sed.

Si Aún No Genera Standalone

Error posible:

 ERROR: standalone output no generado

Causa: Hay un error en el build de Next.js que impide generar el standalone

Solución: Revisar los logs del build de Next.js (línea que dice “npm run build”)



Lecciones Aprendidas

1. Variables de Entorno en Docker

Las variables ENV en Dockerfile no siempre se leen correctamente en todos los contextos. Es más confiable modificar archivos directamente cuando sea posible.

2. sed es Poderoso

`sed` es una herramienta excelente para modificar archivos durante el build de Docker sin cambiar el código fuente.

3. Verificación es Clave

Agregar `grep` después del `sed` nos permite verificar que el cambio se aplicó correctamente.

4. Debugging Iterativo

Cada fix se basa en la información del anterior:

- v9.1: Agregamos la ENV
- v9.2: Arreglamos npm install
- v9.3: Agregamos logs para ver qué pasaba
- v9.4: Con los logs vimos que standalone no se generaba, entonces forzamos con sed



Checklist de Verificación

Antes de considerar este fix exitoso:

- [x] sed agregado al Dockerfile
- [x] grep para verificar el cambio
- [x] Versión actualizada a 9.4
- [x] Documentación creada
- [] Push a GitHub
- [] Rebuild en EasyPanel
- [] Logs muestran “Configuración aplicada: output: ‘standalone’,”
- [] Build completa sin errores

- [] Standalone output verificado
- [] Aplicación funciona correctamente

Referencias

- [Next.js Standalone Output](https://nextjs.org/docs/advanced-features/output-file-tracing#automatically-copying-traced-files) (https://nextjs.org/docs/advanced-features/output-file-tracing#automatically-copying-traced-files)
- [Docker RUN sed Command](https://docs.docker.com/engine/reference/builder/#run) (https://docs.docker.com/engine/reference/builder/#run)
- [sed Command Tutorial](https://www.gnu.org/software/sed/manual/sed.html) (https://www.gnu.org/software/sed/manual/sed.html)

Para el Futuro

Si necesitas cambiar la configuración de Next.js durante el build de Docker:


```
# Patrón general
RUN sed -i "s/CONFIGURACION_ANTIGUA/CONFIGURACION_NUEVA/" archivo.js && \
    grep "PALABRA_CLAVE" archivo.js # Verificar cambio
```

Esto te permite tener configuraciones diferentes para desarrollo (local) y producción (Docker) sin duplicar archivos.

Versión: 9.4

Mejora: Standalone output forzado con sed

Objetivo: Garantizar generación de .next/standalone

Estado:  Implementado

Fecha: 2025-10-15

Próximo paso: Push y rebuild