

Guía de Uso: Scripts de Verificación y Diagnóstico

Resumen de Scripts Creados

He creado **2 scripts poderosos** que te ayudarán a prevenir y diagnosticar problemas de cache en EasyPanel:

1 `pre-deploy-verification.sh` - Verificación Pre-Deploy

Úsalo **ANTES** de cada push a GitHub

2 `cache-diagnostics.sh` - Diagnóstico de Cache

Úsalo cuando sospeches problemas de cache

Uso Rápido

Escenario 1: Deploy Normal (Sin Problemas)

```
# 1. Verificar ANTES de hacer push
cd /home/ubuntu/escalafin_mvp
./scripts/pre-deploy-verification.sh

# Si todo está verde ✓, continuar con:
git add .
git commit -m "tu mensaje"
git push origin main

# 3. En EasyPanel: Rebuild normal (sin limpiar cache)
```

Salida esperada:

```
✓ TODO CORRECTO - LISTO PARA HACER PUSH Y REBUILD
```

```
✓ Exitosos: 28
△ Advertencias: 0
x Errores: 0
```

Escenario 2: Deploy con Errores de Build

```
# 1. Si el build en EasyPanel falla, diagnosticar:
cd /home/ubuntu/escalafin_mvp
./scripts/cache-diagnostics.sh

# 2. El script te dirá qué está mal. Ejemplos:

# Si dice: "package-lock.json desactualizado"
cd app
npm install
cd ..

# Si dice: "Cambios sin commitear"
git add .
git commit -m "fix: corregir dependencias"
git push origin main

# 3. Verificar nuevamente
./scripts/pre-deploy-verification.sh

# 4. Si ahora todo está verde ✓:
#   → Ve a EasyPanel
#   → Marca "Clear build cache"
#   → Haz clic en "Rebuild"
```

Escenario 3: Cache Antiguo (App Funciona Local pero NO en EasyPanel)

```
# 1. Diagnosticar
./scripts/cache-diagnostics.sh

# 2. Buscar la sección "RESUMEN DEL DIAGNÓSTICO"
#   Si dice: "Se detectaron N problema(s)"
#   → Lee los problemas detectados arriba

# 3. Corregir cada problema detectado

# 4. Verificar que todo esté correcto
./scripts/pre-deploy-verification.sh

# 5. Push a GitHub (si había cambios)
git push origin main

# 6. En EasyPanel:
#   ✓ Clear build cache (OBLIGATORIO)
#   → Rebuild
```



Qué Verifica Cada Script

Script 1: pre-deploy-verification.sh

✓ Verifica:

1. Archivos Críticos

- ✓ package.json existe
- ✓ package-lock.json existe y está sincronizado
- ✓ Dockerfile existe
- ✓ docker-compose.yml existe
- ✓ schema.prisma existe

2. Scripts de Producción

- ✓ start-improved.sh existe
- ✓ emergency-start.sh existe
- ✓ healthcheck.sh existe
- ⚠️ setup-users-production.js (opcional pero recomendado)

3. Directorios Esenciales

- ✓ app/ existe
- ✓ app/components/ existe
- ✓ app/lib/ existe
- ✓ app/api/ existe
- ✓ app/prisma/ existe

4. Contenido de Dockerfile

- ✓ WORKDIR /app configurado
- ✓ package-lock.json referenciado
- ✓ Scripts de inicio copiados

5. Archivo .dockerignore

- ✓ Scripts de inicio NO están ignorados
- ✓ Archivos de producción NO están ignorados

6. Sincronización de Dependencias

- ✓ Google Drive (googleapis) en package-lock.json
- ✓ Chatwoot en package-lock.json
- ✓ Todas las dependencias sincronizadas

7. Estado del Repositorio Git

- ✓ No hay cambios sin commitear
- ✓ No hay commits sin hacer push
- ✓ Rama actual es main

8. Permisos de Scripts

- ✓ Todos los scripts tienen permisos de ejecución



Códigos de Salida:

- 0 = Todo perfecto, listo para deploy
- 1 = Advertencias menores, revisar pero puede continuar
- 2 = Errores críticos, NO hacer push hasta corregir

Script 2: `cache-diagnostics.sh`

Diagnostica:

1. Timestamps de Archivos

- Compara fechas de `Dockerfile`, `package.json`, `package-lock.json`
- Detecta si `package-lock.json` es más antiguo que `package.json`
- **Problema común:** Si `package.json` cambió pero `package-lock.json` no, causará error en EasyPanel

2. Sincronización con GitHub

- Verifica último commit local vs GitHub
- Detecta commits sin hacer push
- **Problema común:** Cambios locales que no llegaron a GitHub

3. Cambios sin Commitear

- Lista archivos críticos modificados pero sin commitear
- **Problema común:** Modificaste `Dockerfile` pero olvidaste hacer commit

4. Coherencia de Dockerfile

- Verifica que todos los archivos que el `Dockerfile` copia existan
- **Problema común:** `Dockerfile` referencia `start-improved.sh` pero el archivo no existe

5. Síntomas de Cache Antiguo

- Calcula cuánto tiempo pasó desde el último commit
- Si fue hace más de 1 hora, sugiere verificar

6. Hashes de Verificación

- Genera hashes MD5 de archivos críticos
- Puedes comparar estos hashes con lo que EasyPanel está usando

Salida Final:

RESUMEN DEL DIAGNÓSTICO

✓ NO se detectaron problemas de cache

O si hay problemas:

⚠ Se detectaron 3 problema(s) que pueden causar cache antiguo

Acciones recomendadas:

1. Corrige los problemas indicados arriba
2. Haz commit y push de todos los cambios
3. En EasyPanel: Clear build cache + Rebuild

Casos de Uso Reales

Caso 1: Agregaste Nueva Dependencia

```
# 1. Instalaste nueva dependencia
cd app
npm install nueva-dependencia
cd ..

# 2. Verificar que se sincronizó
./scripts/pre-deploy-verification.sh

# Debe mostrar:
# ✓ package-lock.json está sincronizado

# 3. Commit y push
git add app/package.json app/package-lock.json
git commit -m "feat: agregar nueva-dependencia"
git push origin main

# 4. Deploy normal en EasyPanel
```

Caso 2: Modificaste el Dockerfile

```
# 1. Después de modificar Dockerfile
./scripts/pre-deploy-verification.sh

# Debe mostrar:
# ✓ WORKDIR configurado correctamente
# ✓ Scripts de inicio copiados en Dockerfile

# 2. Si todo está verde ✓
git add Dockerfile
git commit -m "fix: actualizar Dockerfile"
git push origin main

# 3. En EasyPanel:
#   → Clear build cache (OBLIGATORIO al cambiar Dockerfile)
#   → Rebuild
```

Caso 3: EasyPanel Dice “archivo no encontrado” pero el archivo existe

```
# 1. Diagnosticar
./scripts/cache-diagnostics.sh

# Posibles causas detectadas:
# △ Hay cambios sin commitear
# △ Hay commits sin hacer push
# △ Archivo está en .dockerignore

# 2. Corregir según lo detectado
git add .
git commit -m "fix: incluir archivos faltantes"
git push origin main

# 3. En EasyPanel:
#   ✓ Clear build cache
#   → Rebuild
```

Interpretación de Mensajes

Mensajes Verdes (✓) - Todo Bien

- ☒ **package.json** principal
- ☒ **package-lock.json** sincronizado
- ☒ Dockerfile principal
- ☒ Scripts de inicio copiados en Dockerfile

Acción: ¡Ninguna! Todo está perfecto.

Mensajes Amarillos (△) - Advertencias

- ☐ setup-users-production.js - RECOMENDADO
- ☐ Hay cambios sin commitear
- ☐ start-improved.sh NO tiene permisos de ejecución




Acción: Revisar pero no es crítico. Puede continuar si entiendes el riesgo.

Corrección ejemplo:

```
# Para permisos:
chmod +x start-improved.sh

# Para cambios sin commitear:
git add .
git commit -m "mensaje"
```

Mensajes Rojos (X) - Errores Críticos

-  **package**-lock.json sincronizado - FALTA
-  Scripts de inicio NO copiados en Dockerfile
-  Google Drive: falta en **package**-lock.json

Acción: OBLIGATORIO corregir antes de hacer deploy.

Corrección ejemplo:

```
# Para dependencias desincronizadas:
cd app
npm install
cd ..

# Verificar nuevamente:
./scripts/pre-deploy-verification.sh
```



Buenas Prácticas



SIEMPRE:

1. **Ejecuta verificación ANTES de push**

```
bash
./scripts/pre-deploy-verification.sh
```

2. **Si hay errores rojos (X), corrígelos primero**

3. **Después de instalar dependencias, verifica**

```
bash
cd app && npm install && cd ..
./scripts/pre-deploy-verification.sh
```

4. **Si modificas Dockerfile, limpia cache en EasyPanel**



NUNCA:

1. **No hagas push sin verificar primero**

- Causa: Problemas en EasyPanel que podrías haber evitado

2. **No ignores errores rojos (X)**

- Causa: Build fallará en EasyPanel 100%

3. **No olvides `npm install` después de cambiar `package.json`**

- Causa: package-lock.json desactualizado → build falla

4. **No modifiques `.dockerignore` sin verificar**

- Causa: Archivos críticos pueden ser ignorados



Flujo de Trabajo Completo (Recomendado)

```
# =====
#  ANTES DE CADA DEPLOY
#  =====

# 1. Verificar estado actual
cd /home/ubuntu/escalafin_mvp
./scripts/pre-deploy-verification.sh

# 2. Si hay errores, corregirlos
# ... (según lo que indique el script)

# 3. Verificar nuevamente
./scripts/pre-deploy-verification.sh

# 4. Si todo está verde ✓
git add .
git commit -m "tu mensaje descriptivo"
git push origin main

# 5. En EasyPanel:
#   - Deploy normal: Rebuild
#   - Si cambios en Dockerfile: Clear cache + Rebuild

# =====
#  SI HAY PROBLEMAS EN EASYPANEL
#  =====

# 1. Diagnosticar el problema
./scripts/cache-diagnostics.sh

# 2. Leer el RESUMEN DEL DIAGNÓSTICO

# 3. Corregir problemas detectados

# 4. Verificar que se corrigieron
./scripts/pre-deploy-verification.sh

# 5. Push si es necesario
git push origin main

# 6. En EasyPanel:
#   ✓ Clear build cache
#   → Rebuild
```



Solución de Problemas

Problema: Script dice “archivo no encontrado”

Causa: Ruta incorrecta

Solución:

```
cd /home/ubuntu/escalafin_mvp
./scripts/pre-deploy-verification.sh
```

Problema: “Permission denied” al ejecutar script

Causa: Sin permisos de ejecución

Solución:

```
chmod +x scripts/pre-deploy-verification.sh
chmod +x scripts/cache-diagnostics.sh
```

Problema: Script muestra muchos errores rojos (X)

Causa: Archivos críticos faltan o están mal configurados

Solución:

1. Lee cada error rojo cuidadosamente
2. Corrige uno por uno
3. Ejecuta el script nuevamente después de cada corrección
4. Cuando todos estén verdes ✓, haz push

Problema: “package-lock.json desactualizado”

Causa: Modificaste package.json pero no ejecutaste `npm install`

Solución:

```
cd app
npm install
cd ..
git add app/package-lock.json
git commit -m "fix: actualizar package-lock.json"
git push origin main
```



Documentación Adicional

Archivos Relacionados:

- `GUIA_LIMPIAR_CACHE_EASYPANEL.md` - Guía completa para limpiar cache
 - `COMANDOS_UTILES_CACHE.md` - Comandos útiles de Git, Docker, etc.
 - `scripts/pre-deploy-verification.sh` - Script de verificación
 - `scripts/cache-diagnostics.sh` - Script de diagnóstico
-

✓ Checklist de Uso

Antes de CADA deploy:

```

❏ cd /home/ubuntu/escalafin_mvp
❏ ./scripts/pre-deploy-verification.sh
❏ Revisar salida - TODO debe estar verde ✓
❏ Si hay errores rojos ✗, corregirlos
❏ git add . && git commit && git push

```

Después de cambios en dependencias:

```

❏ cd app && npm install
❏ cd .. && ./scripts/pre-deploy-verification.sh
❏ Verificar: "✓ package-lock.json sincronizado"
❏ git add . && git commit && git push

```

Si hay problemas en EasyPanel:

```

❏ ./scripts/cache-diagnostics.sh
❏ Leer RESUMEN DEL DIAGNÓSTICO
❏ Corregir problemas detectados
❏ ./scripts/pre-deploy-verification.sh
❏ En EasyPanel: Clear cache + Rebuild

```

🎯 Resumen Final

Los 2 scripts creados:

1. **pre-deploy-verification.sh**
 - ✓ Verifica 28+ puntos críticos
 - ✓ Detecta problemas ANTES de push
 - ✓ Ejecutar SIEMPRE antes de deploy
2. **cache-diagnostics.sh**
 - 🔍 Diagnostica problemas de cache
 - 🔍 Detecta desincronizaciones
 - 🔍 Ejecutar cuando hay errores en EasyPanel

¡Con estos scripts, los problemas de cache serán cosa del pasado!

☎ Soporte

Si un script muestra un error que no entiendes:

1. **Lee el mensaje completo** - Siempre explica qué está mal
2. **Busca en esta guía** - Casos comunes están documentados
3. **Revisa los archivos de documentación** relacionados

Guía actualizada: 29 de Octubre, 2025

Proyecto: EscalaFin MVP - Sistema de Gestión de Préstamos