






Verificación GitHub Actions - Dockerfile v11.0

CAMBIOS APLICADOS Y PUSHED

Commit Realizado

Commit: ab50611
Mensaje: fix: unificar Dockerfile v11.0 para GitHub Actions y Coolify
Estado:  Pushed a GitHub

Archivos Actualizados

-  Dockerfile (v10.0 → v11.0)
-  .github/workflows/ci.yml (yarn → npm)
-  Dockerfile.v10.backup (creado)
-  FIX_GITHUB_ACTIONS_DOCKERFILE.md (creado)

GITHUB ACTIONS SE DISPARARÁ AUTOMÁTICAMENTE

¿Qué Sucederá Ahora?

Después del push, GitHub Actions automáticamente:

1. **Detecta el push a** `main`
2. **Inicia el workflow:** “Build and Push Docker Image”
3. **Ejecuta el build** con el nuevo Dockerfile v11.0
4. **Pushea la imagen** a Docker Hub

Tiempo estimado: 10-15 minutos

CÓMO VERIFICAR EL BUILD

Paso 1: Ir a GitHub Actions

1. **Abrir en tu navegador:**
`https://github.com/qhosting/escalafin-mvp/actions`
2. **Buscar el workflow más reciente:**
 - Debería estar en progreso (amarillo 🟡)
 - Nombre: “Build and Push Docker Image”
 - Commit: `ab50611` - “fix: unificar Dockerfile v11.0...”

Paso 2: Ver Logs en Vivo

1. **Click en el workflow en progreso**
2. **Click en el job:** "build-and-push"
3. **Expandir cada paso** para ver detalles:
 - Checkout repository
 - Set up Docker Buildx
 - Log in to Docker Hub
 - **Build and push Docker image** ← ESTE ES EL IMPORTANTE
4. **En el paso "Build and push", buscar:**

✅ Lo que DEBERÍAS ver:

...

#8 [deps 3/3] RUN echo "=== Instalando dependencias con NPM ===" && ...

#8 === Instalando dependencias con NPM ===

#8 Limpiando cache...

#8 Instalando todas las dependencias (dev + prod)...

#8 ✅ Dependencias instaladas correctamente

#10 [builder 4/4] RUN echo "=== Building Next.js ===" && ...

#10 === Building Next.js ===

#10 > next build

#10 ✓ Creating an optimized production build

#10 ✓ Compiled successfully

#10 ✅ Build completado

...

❌ Lo que NO deberías ver:

Error: ENOENT: no such file or directory, open 'yarn.lock'

yarn: not found

Failed to install dependencies with yarn

Paso 3: Verificar Build Exitoso

Cuando el workflow complete:

1. **Icono debe cambiar a:** ✅ Verde (exitoso)
2. **Mensaje final:**

```
successfully pushed to docker.io/qhosting/escalafin-mvp:main
successfully pushed to docker.io/qhosting/escalafin-mvp:latest
successfully pushed to docker.io/qhosting/escalafin-mvp:main-ab50611
```
3. **Duración esperada:** ~10-15 minutos



VERIFICAR IMAGEN EN DOCKER HUB

Paso 1: Ir a Docker Hub

1. **Abrir en tu navegador:**

```
https://hub.docker.com/r/qhosting/escalafin-mvp/tags
```

2. Verificar nuevos tags:

- latest (actualizado recientemente)
- main (actualizado recientemente)
- main-ab50611 (nuevo tag)

Paso 2: Verificar Metadata

En Docker Hub, cada imagen debe mostrar:

```
Platforms: linux/amd64, linux/arm64
Last pushed: hace X minutos
Size: ~600-800 MB (dependiendo de optimización)
```

Paso 3: Verificar Labels

Los labels deben incluir:

```
org.opencontainers.image.version: main
org.opencontainers.image.revision: ab50611...
org.opencontainers.image.source: https://github.com/qhosting/escalafin-mvp
org.opencontainers.image.licenses: MIT
```



LOGS DETALLADOS - QUÉ BUSCAR

Sección: Build and push Docker image

Stage 1: deps

```
#8 [deps 3/3] RUN echo "=== Instalando dependencias con NPM ===" && ...
#8 0.389 === Instalando dependencias con NPM ===
#8 0.390 Limpiando cache...
#8 1.234 Instalando todas las dependencias (dev + prod)...
#8 45.678 added 1234 packages in 44s
#8 45.890 [✓] Dependencias instaladas correctamente
#8 DONE 46.0s
```

✓ Indicadores de éxito:

- "Instalando dependencias con NPM"
- "added XXX packages"
- "Dependencias instaladas correctamente"

✗ Errores a buscar (NO deberían aparecer):

- "ENOENT: no such file or directory, open 'yarn.lock'"
- "yarn: not found"
- "npm ERR!"

Stage 2: builder

```
#10 [builder 3/4] RUN echo "=== Generando Prisma Client ===" && ...
#10 0.234 === Generando Prisma Client ===
#10 2.567 ✓ Prisma Client generado
#10 DONE 2.8s

#11 [builder 4/4] RUN echo "=== Building Next.js ===" && ...
#11 0.123 === Building Next.js ===
#11 0.234
#11 1.456 > next build
#11 3.789
#11 5.012 ✓ Creating an optimized production build
#11 87.345 ✓ Compiled successfully
#11 92.456 ✓ Collecting page data
#11 95.678 ✓ Generating static pages (42/42)
#11 96.123 ✓ Collecting build traces
#11 97.234 ✓ Finalizing page optimization
#11 97.890 ✓ Build completado
#11 DONE 98.0s
```

✓ Indicadores de éxito:

- "Compiled successfully"
- "Generating static pages"
- "Build completado"

Stage 3: Export to registry

```
#15 exporting to image
#15 exporting layers 45.6s done
#15 exporting manifest sha256:abc123... 0.1s done
#15 exporting config sha256:def456... 0.0s done
#15 pushing layers
#15 pushing layers 120.3s done
#15 DONE 165.9s
```

✓ Indicadores de éxito:

- "exporting layers"
- "pushing layers"
- "DONE" sin errores

RESULTADOS ESPERADOS

Build Exitoso

```
✓ Stage deps: ~45 segundos
✓ Stage builder: ~100 segundos
✓ Stage runner: ~5 segundos
✓ Export to registry: ~160 segundos


---


⌚ Total: ~10-15 minutos
✓ Status: Success
```

Imagen en Docker Hub

- ✓ Tag: latest (updated)
- ✓ Tag: main (updated)
- ✓ Tag: main-ab50611 (new)
- ✓ Platforms: linux/amd64, linux/arm64
- ✓ Size: ~600-800 MB

TROUBLESHOOTING

Si GitHub Actions Falla

Error: “yarn.lock not found”

Causa: GitHub Actions está usando un Dockerfile en caché (no el nuevo)

Solución:

1. Ir a Actions **tab**
2. Click en **"Re-run all jobs"**
3. O esperar al próximo commit

Error: “npm install failed”

Causa: Problema de red o dependencia específica

Solución:

1. Ver **logs** detallados del error
2. Verificar que package.json esté correcto
3. Puede ser temporal, reintentar

Error: “Build timed out”

Causa: Runner de GitHub Actions con pocos recursos



Solución:

1. Es temporal, reintentar workflow
2. O esperar a que GitHub libere recursos

Error: “Cannot push to Docker Hub”

Causa: Credenciales de Docker Hub incorrectas o expiradas

Solución:

1. Verificar secrets en GitHub:
Settings  Secrets **and** variables  Actions
2. Verificar:
 - DOCKERHUB_USERNAME
 - DOCKERHUB_TOKEN
3. Regenerar **token** si es necesario

Si el Build es Muy Lento

Normal:

- Primera vez después de cambios: ~15 minutos
- Builds subsecuentes (con caché): ~8-10 minutos


Muy lento (>20 minutos):

- Problema de red en GitHub runners
- Demasiadas dependencias sin caché
- Solución: Esperar o reintentar más tarde

CHECKLIST DE VERIFICACIÓN

Use esta lista para verificar que todo funcione:

GitHub Actions

- ☐ Workflow “Build and Push Docker Image” inició
- ☐ Está usando Dockerfile (no Dockerfile.old o similar)
- ☐ Logs muestran “Instalando dependencias con NPM”
- ☐ No hay errores de yarn.lock
- ☐ Stage deps completa sin errores
- ☐ Stage builder completa sin errores
- ☐ Next.js compila exitosamente
- ☐ Export to registry completa
- ☐ Workflow completa con  verde

Docker Hub

- ☐ Tag `latest` actualizado
- ☐ Tag `main` actualizado
- ☐ Tag `main-ab50611` creado
- ☐ Platforms: linux/amd64, linux/arm64
- ☐ Size razonable (~600-800 MB)

Coolify (Después)

- ☐ Re-deploy en Coolify
- ☐ Build completa sin errores
- ☐ Aplicación inicia correctamente
- ☐ No hay errores de yarn.lock en logs

ENLACES ÚTILES

GitHub

- **Actions:** <https://github.com/qhosting/escalafin-mvp/actions>
- **Último commit:** <https://github.com/qhosting/escalafin-mvp/commit/ab50611>

- **Workflow file:** <https://github.com/qhosting/escalafin-mvp/blob/main/.github/workflows/docker-build.yml>

Docker Hub

- **Repository:** <https://hub.docker.com/r/qhosting/escalafin-mvp>
- **Tags:** <https://hub.docker.com/r/qhosting/escalafin-mvp/tags>

Coolify

- **Dashboard:** <https://adm.escalafin.com>



NOTAS ADICIONALES

Workflow Automático

El workflow `docker-build.yml` se dispara automáticamente en:

- ☒ Push a `main` branch
- ☒ Manual dispatch (botón en GitHub Actions)

No necesitas hacer nada, GitHub Actions se encarga de todo.

CI Pipeline Separado

También hay un workflow `ci.yml` que:

- Se dispara en push a `main` o `develop`
- Hace tests, type check, build
- También fue actualizado a NPM

Ambos workflows ahora son consistentes (NPM).

Siguientes Deploys

Para futuros cambios:

```
# 1. Hacer cambios en el código
# 2. Commit
git add .
git commit -m "tu mensaje"

# 3. Push
git push origin main

# 4. GitHub Actions automáticamente:
#   - Construye nueva imagen
#   - La pushea a Docker Hub
#   - Tags: latest, main, main-<commit>

# 5. En Coolify:
#   - Hacer re-deploy manual
#   - 0 configurar auto-deploy en cada push
```

RESUMEN FINAL

Todo Completado

- [x] Dockerfile v11.0 unificado
- [x] CI workflow actualizado a NPM
- [x] Cambios committed y pushed
- [x] GitHub Actions se disparará automáticamente

En Progreso (Automático)

- [] GitHub Actions construyendo imagen
- [] Pusheando a Docker Hub

Tu Siguiente Acción


OPCIÓN 1: Monitorear (Opcional)

Ir a: <https://github.com/qhosting/escalafin-mvp/actions>

Observar el workflow completarse (~10-15 minutos)

OPCIÓN 2: Esperar y Verificar Después

En ~15 minutos, verificar:

- GitHub Actions tiene  verde
- Docker Hub tiene nuevos tags

OPCIÓN 3: Continuar con Coolify

No necesitas esperar GitHub Actions para hacer re-deploy en Coolify.
Coolify usa el código de GitHub directamente.

Estado Actual:  Push completado - GitHub Actions en progreso

Tiempo estimado: 10-15 minutos para build completo

Próxima verificación: En 15 minutos revisar GitHub Actions
