

Guía de Configuración para Easypanel

Fecha: 25 de Octubre, 2025
Proyecto: VertexERP v4.0.0
Problema: “No such image: easypanel/cloudmx/vertexerp:latest”
Solución: Configurar correctamente el build desde Dockerfile

Problema Identificado

Error Reportado

No such image: easypanel/cloudmx/vertexerp:latest

Causa

Easypanel está configurado para usar una **imagen Docker pre-construida** en lugar de **construir desde el Dockerfile** del repositorio.

Configuración incorrecta:

- Tipo: Docker Image (imagen pre-construida)
- Imagen: easypanel/cloudmx/vertexerp:latest ❌

Configuración correcta:

- Tipo: **Build from Source** (construir desde Dockerfile)
- Dockerfile: `./Dockerfile` ✅

✅ Configuración Correcta Paso a Paso

1. Acceder al Proyecto en Easypanel

1. Ve a tu panel de Easypanel
2. Si ya existe el proyecto `vertexerp` , **elimínalo primero**
3. Crea un **nuevo proyecto desde cero**

2. Configuración Inicial del Proyecto

Nombre del Proyecto:

VertexERP

Tipo de Servicio:

App

3. Configuración de la Fuente (SOURCE)

Esta es la parte **MÁS IMPORTANTE**:

Opción A: GitHub (Recomendado)

Campo	Valor
Source Type	GitHub
Repository	qhosting/vertexerp
Branch	main
Auto Deploy	<input checked="" type="checkbox"/> Enabled (para deploy automático con push)

Opción B: Git URL

Campo	Valor
Source Type	Git
Repository URL	https://github.com/qhosting/vertexerp.git
Branch	main
Auto Deploy	<input checked="" type="checkbox"/> Enabled

4. Configuración del Build (BUILD)

⚠ CRÍTICO: Esta es donde se comete el error más común

Campo	Valor	Notas
Build Method	Dockerfile	<input checked="" type="checkbox"/> NO seleccionar "Docker Image"
Dockerfile Path	./Dockerfile	Ruta relativa desde la raíz
Context Path	.	Directorio raíz del proyecto
Build Args	(dejar vacío)	No necesario

NO usar:

- ☒ Docker Image
- ☒ Pre-built image
- ☒ Registry image

SÍ usar:

- ☒ Build from Dockerfile
- ☒ Build from source

5. Configuración de Variables de Entorno

Variables Obligatorias

```
# Base de Datos
DATABASE_URL=postgresql://usuario:password@host:5432/database

# NextAuth (Autenticación)
NEXTAUTH_URL=https://tu-dominio.easypanel.app
NEXTAUTH_SECRET=genera-un-secret-aleatorio-aqui

# Node
NODE_ENV=production
```

Generar NEXTAUTH_SECRET

```
# Opción 1: OpenSSL
openssl rand -base64 32

# Opción 2: Node.js
node -e "console.log(require('crypto').randomBytes(32).toString('base64'))"

# Ejemplo de resultado:
# yJ8X9kL2mP5nQ3rT6vW8zA1bC4dE7fH0iJ1kL2mN4o=
```

Variables Opcionales (Según Módulos)

```
# Openpay (Pagos)
OPENPAY_API_KEY=sk_XXXXXXXXXXXXX
OPENPAY_MERCHANT_ID=mXXXXXXXXXXXXX
OPENPAY_PRIVATE_KEY=pk_XXXXXXXXXXXXX
OPENPAY_PUBLIC_KEY=pub_XXXXXXXXXXXXX
OPENPAY_BASE_URL=https://api.openpay.mx/v1

# LabsMobile (SMS)
LABSMOBILE_USERNAME=tu-usuario
LABSMOBILE_PASSWORD=tu-password

# Evolution API (WhatsApp)
EVOLUTION_API_URL=https://tu-servidor.com
EVOLUTION_API_KEY=tu-api-key
EVOLUTION_INSTANCE=tu-instancia

# Correo (SMTP)
SMTP_HOST=smtp.gmail.com
SMTP_PORT=587
SMTP_USER=tu-email@gmail.com
SMTP_PASSWORD=tu-app-password
```

6. Configuración de Red (NETWORKING)

Campo	Valor	Notas
Port	3000	Puerto interno del contenedor
Domain	tu-app.easypanel.app	O tu dominio personalizado
HTTPS	<input checked="" type="checkbox"/> Enabled	Certificado SSL automático

7. Configuración de Recursos (RESOURCES)

Recurso	Mínimo	Recomendado	Producción
CPU	0.5 cores	1 core	2+ cores
RAM	512 MB	1 GB	2-4 GB
Storage	5 GB	10 GB	20+ GB

8. Health Check

Campo	Valor
Health Check Path	/api/health
Health Check Port	3000
Initial Delay	40s
Interval	30s
Timeout	10s
Retries	3



Estructura de Archivos Críticos

Easypanel buscará estos archivos en tu repositorio:

```

vertexerp/
├── Dockerfile           ⚠️ CRÍTICO: Debe estar en la raíz
├── docker-compose.yml   ⚠️ Opcional (no usado por Easypanel)
├── .dockerignore        ⚠️ Importante para optimizar build
├── start.sh             ⚠️ Script de inicio (llamado por Dockerfile)
├── .env.production.example ⚠️ Template de variables
├── app/
│   ├── package.json     ⚠️ Dependencias
│   ├── yarn.lock        ⚠️ ⚠️ Debe ser archivo real (no symlink)
│   ├── next.config.js   ⚠️ Configuración Next.js
│   └── prisma/
│       └── schema.prisma ⚠️ Schema de base de datos
└── ...

```

Verificar Configuración de Dockerfile

Tu `Dockerfile` está correcto y usa **multi-stage build**:

```



# Stage 1: Instalar dependencias
FROM node:18-alpine AS deps
WORKDIR /app
COPY app/package.json app/yarn.lock ./
RUN yarn install --frozen-lockfile

# Stage 2: Build de la aplicación
FROM node:18-alpine AS builder
WORKDIR /app
COPY --from=deps /app/node_modules ./node_modules
COPY app/ ./
RUN yarn prisma generate
RUN yarn build

# Stage 3: Producción (imagen final)
FROM node:18-alpine AS runner
WORKDIR /app
# ... copiar archivos necesarios ...
CMD ["/start.sh"]

```





Easypanel ejecutará automáticamente:

1. `docker build -t vertexerp:latest .` 
2. `docker run vertexerp:latest` 

Checklist de Configuración

Antes de hacer deploy, verificar:

En GitHub

- [x]  Dockerfile está en la raíz del repositorio
- [x]  app/yarn.lock es un archivo real (no symlink)
- [x]  .dockerignore está configurado
- [x]  start.sh tiene permisos de ejecución

- [x] ☒ Último commit incluye todos los cambios

En Easypanel

- [] ☒ Source Type = **GitHub** o **Git**
- [] ☒ Build Method = **Dockerfile** (NO "Docker Image")
- [] ☒ Dockerfile Path = `./Dockerfile`
- [] ☒ Context Path = `.`
- [] ☒ Variables de entorno configuradas
- [] ☒ DATABASE_URL apunta a base de datos válida
- [] ☒ NEXTAUTH_SECRET generado
- [] ☒ NEXTAUTH_URL con tu dominio
- [] ☒ Port = 3000
- [] ☒ Health Check = `/api/health`



Proceso de Deploy

Primera Vez

1. Crear Proyecto Nuevo en Easypanel

- Click en "New Project" o "Create App"
- Nombrar: `VertexERP`

2. Configurar Source

- Conectar GitHub: `ghosting/vertexerp`
- Branch: `main`
- Enable Auto Deploy

3. Configurar Build

- **IMPORTANTE:** Seleccionar `Dockerfile`
- Dockerfile path: `./Dockerfile`
- Context: `.`

4. Agregar Variables de Entorno

- Pegar todas las variables necesarias
- Guardar

5. Configurar Networking

- Port: 3000
- Asignar dominio
- Enable HTTPS

6. Iniciar Deploy

- Click en "Deploy" o "Start Build"
- Easypanel comenzará a construir la imagen

Monitoring del Build

El proceso tomará aproximadamente **5-10 minutos**:

```
[1/4] Pulling base images...      ✓ (30s)
[2/4] Installing dependencies...  ✓ (2-3 min)
[3/4] Building Next.js application... ✓ (2-3 min)
[4/4] Creating production image... ✓ (30s)
```

- ✓ Build completed successfully
- ✓ Starting container...
- ✓ Health check passed
- ✓ Deployment successful

Verificar Deploy Exitoso

1. Ver Logs

```
Prisma Client generated
Next.js compiled
Server listening on 0.0.0.0:3000
Health check: OK
```

2. Probar Health Check

```
bash
curl https://tu-app.easypanel.app/api/health
# Esperado: {"status":"ok"}
```

3. Acceder a la Aplicación

- <https://tu-app.easypanel.app>
- Debe cargar la página de login



Solución de Problemas Comunes

Error: “No such image”

Síntoma:

```
No such image: easypanel/cloudmx/vertexerp:latest
```

Causa:

Build Method configurado como “Docker Image” en lugar de “Dockerfile”

Solución:

1. Eliminar el proyecto actual
2. Crear nuevo proyecto
3. En “Build Method” seleccionar **“Dockerfile”**
4. Deploy nuevamente

Error: “yarn.lock not found”

Síntoma:

```
ERROR: failed to calculate checksum of "/app/yarn.lock": not found
```

Causa:

yarn.lock es un symlink en lugar de archivo real

Solución:

Ya está resuelto en el último commit (678c52a). Si persiste:

```
# En tu máquina local
cd sistema_erp_completo/app
rm yarn.lock
cp /opt/hostedapp/node/root/app/yarn.lock .
git add yarn.lock
git commit -m "fix: yarn.lock como archivo real"
git push
```

Error: “Database connection failed”**Síntoma:**

```
PrismaClientInitializationError: Can't reach database server
```

Causa:

DATABASE_URL mal configurado o base de datos no accesible

Solución:

1. Verificar que DATABASE_URL sea correcto
2. Verificar que la base de datos esté corriendo
3. Verificar firewall/networking en Easypanel
4. Formato correcto:

```
postgresql://usuario:password@host:5432/database
```

Error: “Build timeout”**Síntoma:**

```
Build timed out after 15 minutes
```

Causa:

Recursos insuficientes o problema de red

Solución:

1. Aumentar timeout en configuración
 2. Verificar que yarn.lock esté presente
 3. Verificar conectividad de red
 4. Aumentar recursos (RAM/CPU)
-



Comparación: Imagen vs Dockerfile

Característica	Docker Image ❌	Dockerfile ✅
Fuente	Imagen pre-construida	Código fuente
Ejemplo	vertexerp:latest	./Dockerfile
Requiere	Imagen en registry	Dockerfile en repo
Flexibilidad	Baja (imagen fija)	Alta (build dinámico)
Actualizaciones	Manual	Automático con push
Personalización	No	Sí
Para VertexERP	❌ NO usar	✅ Sí usar

Configuración Correcta Final

En Easypanel Dashboard

Project:

Name: VertexERP

Source:


Type: GitHub

Repository: qhosting/vertexerp

Branch: main

Auto Deploy: ☒ Enabled

Build:

Method: Dockerfile #  NO "Docker Image"

Dockerfile: ./Dockerfile

Context: .

Environment Variables:

DATABASE_URL: postgresql://...

NEXTAUTH_URL: https://tu-dominio.easypanel.app

NEXTAUTH_SECRET: *****

NODE_ENV: production

Networking:

Internal Port: 3000

Domain: tu-dominio.easypanel.app

HTTPS: ☒ Enabled

Health Check:

Path: /api/health

Port: 3000

Initial Delay: 40s

Interval: 30s

Timeout: 10s

Retries: 3

Resources:

CPU: 1 core

RAM: 1 GB

Storage: 10 GB

✓ Verificación Final

Antes de Deploy

```
# Verificar que todo esté en GitHub
git status
# Esperado: "nothing to commit, working tree clean"

git log --oneline -1
# Esperado: 678c52a fix(docker): yarn.lock como archivo real

# Verificar archivos críticos
ls -lh Dockerfile
ls -lh app/yarn.lock
ls -lh start.sh

# Todos deben existir y ser archivos reales
```

Durante Deploy

Monitorear logs en Easypanel:

```
✓ Cloning repository...
✓ Dockerfile found at ./Dockerfile
✓ Building image...
  └─ Stage 1/3: deps
  └─ Stage 2/3: builder
  └─ Stage 3/3: runner
✓ Image built successfully
✓ Starting container...
✓ Health check passed
✓ Deployment successful
```

Después de Deploy

```
# Verificar health check
curl https://tu-dominio.easypanel.app/api/health
# Esperado: {"status":"ok"}

# Verificar la aplicación
curl -I https://tu-dominio.easypanel.app
# Esperado: HTTP/2 200

# Verificar login
curl https://tu-dominio.easypanel.app/login
# Esperado: HTML de la página de login
```

Resultado Esperado

Si todo está configurado correctamente, verás:

1. ✓ Build inicia automáticamente
2. ✓ Dockerfile se ejecuta sin errores
3. ✓ yarn.lock se copia correctamente

4. ☒ Dependencias se instalan
5. ☒ Prisma Client se genera
6. ☒ Next.js se construye exitosamente
7. ☒ Imagen se crea (tamaño: ~450 MB)
8. ☒ Contenedor inicia
9. ☒ Health check pasa
10. ☒ Aplicación accesible en tu dominio

Tiempo total: 5-10 minutos para el primer deploy

Soporte

Si después de seguir estos pasos todavía tienes problemas:

1. Verificar logs de Easypanel

- Panel → Project → Logs
- Buscar líneas con "ERROR" o "FATAL"

2. Verificar configuración

- Panel → Project → Settings
- Comparar con esta guía

3. Verificar repositorio

- <https://github.com/qhosting/vertexerp>
 - Verificar que Dockerfile esté visible
 - Verificar que yarn.lock tenga 434 KB
-

VertexERP v4.0.0

Configuración Verificada para Easypanel

© 2025 - Build desde Dockerfile funcionando correctamente