


# QUICK START DEPLOYMENT - SISTEMA ROSSI

 **Deployment en 10 minutos**

 **Tiempo estimado:** 5-10 minutos

 **Resultado:** Sistema funcionando en producción

## REQUISITOS PREVIOS

- ✓ Node.js >= 18.x
- ✓ PostgreSQL >= 13.x
- ✓ Git
- ✓ 2GB RAM mínimo

## DEPLOYMENT EN 5 PASOS

### Paso 1: Clonar Proyecto (30s)

```
git clone [REPOSITORY_URL] rossi-inventory
cd rossi-inventory
```

### Paso 2: Instalar Dependencias (2-3 min)

```
npm install
```

### Paso 3: Configurar Base de Datos (1 min)

```
# Crear base de datos
createdb rossi_inventory

# Configurar .env
cp .env.example .env

# Editar .env con tus datos:
# DATABASE_URL="postgresql://user:pass@localhost:5432/rossi_inventory"
# NEXTAUTH_SECRET="tu-clave-secreta-minimo-32-caracteres"
# NEXTAUTH_URL="https://tu-dominio.com"

# Aplicar schema y datos
npx prisma db push
npm run db:seed
```

## Paso 4: Build y Start (2-3 min)

```
npm run build # 13 segundos aprox
npm start    # Servidor en puerto 3000
```



## ✓ Paso 5: Verificar (30s)

```
# Abrir en navegador
http://localhost:3000

# Login con:
# Email: admin@rossi.com
# Password: admin123
```

## ACCESO INMEDIATO

### URLs del Sistema

-  **Aplicación:** `http://localhost:3000`
-  **Login:** `http://localhost:3000/auth/signin`
-  **Dashboard:** `http://localhost:3000/dashboard`

### Credenciales de Prueba

Usuario	Email	Contraseña	Rol
Admin	admin@rossi.com	admin123	Super Admin
Almacén	almacen@rossi.com	admin123	Warehouse

## DEPLOYMENT CON DOCKER (Alternativo)

### Dockerfile incluido - Un comando:

```
docker-compose up -d
```

### O manualmente:

```
docker build -t rossi-inventory .
docker run -p 3000:3000 rossi-inventory
```

## DEPLOYMENT EN LA NUBE

---

### Vercel (Recomendado)

```
npm i -g vercel
vercel --prod
```

### Railway

```
npm i -g @railway/cli
railway login
railway deploy
```

### Heroku

```
heroku create rossi-inventory
git push heroku main
```

---

## CONFIGURACIÓN RÁPIDA PRODUCCIÓN

---

### Variables de Entorno Mínimas

```
DATABASE_URL="postgresql://..."
NEXTAUTH_SECRET="clave-segura-32-chars-minimo"
NEXTAUTH_URL="https://tu-dominio.com"
```

### SSL y Dominio (Opcional)

```
# Con Nginx
server {
    listen 443 ssl;
    server_name tu-dominio.com;

    ssl_certificate /path/to/cert.pem;
    ssl_certificate_key /path/to/key.pem;

    location / {
        proxy_pass http://localhost:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

---

## TROUBLESHOOTING RÁPIDO

### Error: Cannot connect to database

```
# Verificar PostgreSQL
sudo systemctl status postgresql
sudo systemctl start postgresql

# Verificar .env
echo $DATABASE_URL
```

### Error: Build failed

```
# Limpiar cache
rm -rf .next node_modules
npm install
npm run build
```

### Error: Login no funciona

```
# Verificar secret
echo $NEXTAUTH_SECRET
# Debe tener mínimo 32 caracteres

# Verificar usuarios seed
npm run db:seed
```

## ¡LISTO!

### Tu Sistema de Inventario Rossi está funcionando:

- ✓ **Frontend:** Interfaz moderna y responsive
- ✓ **Backend:** 23 APIs completamente funcionales
- ✓ **Database:** Schema completo con datos de prueba
- ✓ **Security:** Sistema de usuarios y roles
- ✓ **Performance:** Build optimizado para producción

### Próximos Pasos:

1. **Cambiar passwords** de usuarios de prueba
2. **Configurar backup** de base de datos
3. **Personalizar empresa** (nombre, logo, etc.)
4. **Entrenar usuarios** con sistema funcionando

 **Deployment completado en menos de 10 minutos** 

 **¡El Sistema de Inventario Rossi está listo para usar!**