🚀 COMODÍN IA - GUÍA COMPLETA DE DESPLIEGUE Y TRANSFERENCIA

📋 ÍNDICE

🛠️ STACK TECNOLÓGICO

Frontend

- **Next.js 14.2.28** - Framework React con SSR/SSG
- **React 18.2.0** - Biblioteca de UI
- **TypeScript 5.2.2** - Lenguaje de programación tipado
- **Tailwind CSS 3.3.3** - Framework de utilidades CSS
- **Radix UI** - Componentes de UI accesibles

Backend

- **Node.js 18+** - Runtime de JavaScript
- **Next.js API Routes** - Endpoints API
- **NextAuth.js 4.24.11** - Sistema de autenticación
- **Prisma 6.7.0** - ORM y cliente de base de datos

Base de Datos

- **PostgreSQL 15** - Base de datos principal
- **Redis 7** - Cache y gestión de sesiones
- **Prisma Migrations** - Control de versiones de BD

Servicios Externos

- **Evolution API** - Integración con WhatsApp
- **Stripe** - Procesamiento de pagos internacionales
- **MercadoPago** - Procesamiento de pagos LATAM
- **AWS S3** - Almacenamiento de archivos
- **Abacus AI** - Servicios de inteligencia artificial

Infraestructura

- **Docker & Docker Compose** - Containerización
- **Nginx** - Reverse proxy y servidor web
- **Let's Encrypt** - Certificados SSL gratuitos
- **Portainer** - Gestión visual de contenedores (opcional)

🖥️ PRERREQUISITOS DEL SERVIDOR

Especificaciones Mínimas de Hardware

```
CPU: 4 vCPUs (mínimo 2 vCPUs)
RAM: 8GB (mínimo 4GB)
Almacenamiento: 100GB SSD (mínimo 50GB)
Ancho de banda: 100 Mbps up/down
```

Especificaciones Recomendadas para Producción

```
CPU: 8 vCPUs o más
RAM: 16GB o más
Almacenamiento: 200GB+ SSD NVMe
Ancho de banda: 1 Gbps up/down
Respaldo: Configuración RAID o snapshots automáticos
```

Sistema Operativo Soportado

- **Ubuntu 22.04 LTS** (recomendado)
- **Ubuntu 20.04 LTS**
- **Debian 11/12**
- **CentOS Stream 9**
- **RHEL 8/9**

Software Base Requerido

- **Docker Engine 24.0+**
- **Docker Compose 2.20+**
- **Git 2.34+**
- **Curl, wget**
- **UFW o iptables** (firewall)

⚙️ PREPARACIÓN DEL SERVIDOR

1. Actualización del Sistema

```
# Ubuntu/Debian
sudo apt update && sudo apt upgrade -y
sudo apt install -y curl wget git ufw htop nano

# CentOS/RHEL
sudo dnf update -y
sudo dnf install -y curl wget git firewalld htop nano
```

2. Instalación de Docker

```
# Remover versiones anteriores
sudo apt remove docker docker-engine docker.io containerd runc

# Instalar Docker desde repositorio oficial
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

# Agregar usuario al grupo docker
sudo usermod -aG docker $USER

# Instalar Docker Compose
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-com-
pose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

# Verificar instalación
docker --version
docker-compose --version
```

3. Configuración de Firewall

```
# Ubuntu/Debian con UFW
sudo ufw allow ssh
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw --force enable

# CentOS/RHEL con firewalld
sudo systemctl enable firewalld
sudo systemctl start firewalld
sudo firewall-cmd --permanent --add-service=ssh
sudo firewall-cmd --permanent --add-service=http
sudo firewall-cmd --permanent --add-service=https
sudo firewall-cmd --reload
```

4. Configuración de Límites del Sistema

```
# Aumentar límites de archivos abiertos
sudo tee -a /etc/security/limits.conf << EOF
* soft nofile 65535
* hard nofile 65535
root soft nofile 65535
root hard nofile 65535
EOF

# Configurar límites del kernel
sudo tee -a /etc/sysctl.conf << EOF
vm.max_map_count=262144
net.core.rmem_default=262144
net.core.rmem_max=16777216
net.core.wmem_default=262144
net.core.wmem_max=16777216
EOF

sudo sysctl -p
```

🚀 DESPLIEGUE PASO A PASO

Paso 1: Clonar o Transferir el Código

```
# Opción A: Desde repositorio Git
git clone https://tu-repositorio.com/comodin-ia.git
cd comodin-ia

# Opción B: Transferir archivos via SCP
scp -r ./comodin_ia usuario@servidor:/home/usuario/
ssh usuario@servidor
cd comodin_ia
```

Paso 2: Configurar Variables de Entorno

```
# Copiar y configurar el archivo de environment
cp .env.example .env

# Editar configuración (usar nano, vim, o editor preferido)
nano .env
```

Variables Críticas a Configurar:

```
# Base de datos
DATABASE_URL=postgresql://postgres:TU_PASSWORD_SEGURO@postgres:5432/comodin_ia
POSTGRES_PASSWORD=TU_PASSWORD_SEGURO

# Autenticación
NEXTAUTH_URL=https://tu-dominio.com
NEXTAUTH_SECRET=CLAVE_SECRETA_32_CARACTERES_MINIMO

# Pagos
STRIPE_SECRET_KEY=sk_live_...
MERCADO_PAGO_ACCESS_TOKEN=APP_USR-...

# IA
ABACUSAI_API_KEY=tu_clave_aqui
OPENAI_API_KEY=sk-...

# WhatsApp
EVOLUTION_API_KEY=CLAVE_SEGURA_EVOLUTION

# AWS
AWS_BUCKET_NAME=tu-bucket-s3
AWS_REGION=us-west-2
```

Paso 3: Configurar Nginx

```
# Editar nginx.conf para incluir tu dominio
nano nginx.conf

# Cambiar todas las instancias de "tu-dominio.com" por tu dominio real
sed -i 's/tu-dominio.com/tudominio.com/g' nginx.conf
```

Paso 4: Crear Directorios Necesarios

```
# Crear directorios para SSL y backups
mkdir -p ssl backups logs

# Asignar permisos correctos
chmod 755 ssl backups logs
sudo chown -R $USER:$USER ./
```

Paso 5: Construir y Levantar los Servicios

```
# Construir las imágenes Docker
docker-compose build --no-cache

# Levantar servicios de base de datos primero
docker-compose up -d postgres redis

# Esperar a que estén listos (30 segundos)
sleep 30

# Ejecutar migraciones de base de datos
docker-compose exec app npx prisma migrate deploy
docker-compose exec app npx prisma generate

# Seed de datos iniciales (opcional)
docker-compose exec app npx prisma db seed

# Levantar todos los servicios
docker-compose up -d
```

Paso 6: Verificar Estado de los Servicios

```
# Ver estado de todos los contenedores
docker-compose ps

# Ver logs en tiempo real
docker-compose logs -f

# Ver logs de un servicio específico
docker-compose logs app
docker-compose logs postgres
docker-compose logs evolution-api
```

🔒 CONFIGURACIÓN DE SSL/TLS

Opción A: Let's Encrypt con Certbot (Recomendado)

```
# Instalar Certbot
sudo apt install certbot python3-certbot-nginx -y

# Detener nginx temporalmente
docker-compose stop nginx

# Generar certificados
sudo certbot certonly --standalone -d tudominio.com -d www.tudominio.com

# Copiar certificados al directorio del proyecto
sudo cp /etc/letsencrypt/live/tudominio.com/fullchain.pem ./ssl/
sudo cp /etc/letsencrypt/live/tudominio.com/privkey.pem ./ssl/
sudo chown $USER:$USER ./ssl/*

# Reiniciar nginx
docker-compose start nginx
```

Opción B: Certificados Existentes

```
# Copiar tus certificados existentes
cp tu_certificado.crt ./ssl/fullchain.pem
cp tu_clave_privada.key ./ssl/privkey.pem
chmod 600 ./ssl/*
```

Configurar Renovación Automática

```
# Crear script de renovación
sudo tee /usr/local/bin/renovar-ssl.sh << 'EOF'
#!/bin/bash
cd /ruta/a/comodin_ia
docker-compose stop nginx
certbot renew --standalone
cp /etc/letsencrypt/live/tudominio.com/fullchain.pem ./ssl/
cp /etc/letsencrypt/live/tudominio.com/privkey.pem ./ssl/
chown $USER:$USER ./ssl/*
docker-compose start nginx
EOF

sudo chmod +x /usr/local/bin/renovar-ssl.sh

# Configurar cron para renovación automática
sudo crontab -e
# Agregar línea: 0 3 1 * * /usr/local/bin/renovar-ssl.sh
```

✅ VERIFICACIÓN Y PRUEBAS

1. Health Checks de Servicios

```
# Verificar que todos los contenedores estén saludables
docker-compose ps

# Debe mostrar "healthy" en todos los servicios
# Si alguno muestra "unhealthy", revisar logs:
docker-compose logs [nombre-servicio]
```

2. Pruebas de Conectividad

```
# Probar conectividad a la aplicación
curl -I https://tudominio.com
# Debe retornar: HTTP/2 200

# Probar API de salud
curl https://tudominio.com/api/health
# Debe retornar: {"status":"ok"}

# Probar Evolution API
curl https://tudominio.com/evolution/
# Debe retornar respuesta JSON
```

3. Pruebas de Funcionalidad

1. **Registro de Usuario**: Crear cuenta nueva
2. **Login/Logout**: Verificar autenticación
3. **Carga de Archivos**: Subir imagen de perfil
4. **WhatsApp**: Conectar dispositivo via QR
5. **Pagos**: Realizar transacción de prueba
6. **IA**: Enviar consulta y verificar respuesta

4. Monitoreo de Performance

```
# Monitorear uso de recursos
docker stats

# Ver logs de aplicación
docker-compose logs -f app | grep -E "(ERROR|WARN|performance)"

# Verificar uso de base de datos
docker-compose exec postgres psql -U postgres -d comodin_ia -c "SELECT count(*) FROM
\"User\";"
```

🔧 MANTENIMIENTO REGULAR

Rutinas Diarias

```
#!/bin/bash
# Script: mantenimiento-diario.sh

# Verificar estado de servicios
echo "=== Estado de Servicios ==="
docker-compose ps

# Verificar uso de disco
echo "=== Uso de Disco ==="
df -h

# Limpiar logs antiguos (mantener últimos 7 días)
find ./logs -name "*.log" -mtime +7 -delete

# Verificar backups
echo "=== Últimos Backups ==="
ls -la ./backups/ | tail -5
```

Rutinas Semanales

```
#!/bin/bash
# Script: mantenimiento-semanal.sh

# Actualizar imágenes Docker
echo "=== Actualizando Imágenes Docker ==="
docker-compose pull

# Limpiar recursos no utilizados
echo "=== Limpieza de Docker ==="
docker system prune -f
docker volume prune -f

# Verificar certificados SSL (aviso si expiran en 30 días)
echo "=== Verificación SSL ==="
openssl x509 -in ./ssl/fullchain.pem -text -noout | grep "Not After"

# Analizar logs de errores
echo "=== Análisis de Logs ==="
docker-compose logs --since 168h app | grep -i error | wc -l
```

Rutinas Mensuales

```
#!/bin/bash
# Script: mantenimiento-mensual.sh

# Backup completo de configuraciones
echo "=== Backup de Configuraciones ==="
tar -czf "backup-config-$(date +%Y%m%d).tar.gz" \
    .env docker-compose.yml nginx.conf ssl/

# Análisis de performance de base de datos
echo "=== Análisis de DB Performance ==="
docker-compose exec postgres psql -U postgres -d comodin_ia -c "
SELECT schemaname,tablename,attname,n_distinct,correlation FROM pg_stats
WHERE schemaname = 'public' ORDER BY tablename, attname;
"

# Verificar integridad de backups
echo "=== Verificación de Backups ==="
latest_backup=$(ls -t ./backups/*.sql.gz | head -1)
if [ -f "$latest_backup" ]; then
    gunzip -t "$latest_backup" && echo "Backup OK" || echo "Backup CORRUPTO"
fi
```

🌐 SERVICIOS EXTERNOS

1. Configuración de Stripe

1. **Crear cuenta en Stripe**: https://dashboard.stripe.com
2. **Obtener claves API**:
   - Ir a “Developers > API keys”
   - Copiar “Publishable key” y “Secret key”
   - Para webhooks: crear endpoint `https://tudominio.com/api/webhooks/stripe`

3. **Configurar Webhooks**:
```
Eventos a escuchar:
   - invoice.payment_succeeded
   - invoice.payment_failed
   - customer.subscription.created
   - customer.subscription.updated
   - customer.subscription.deleted
```

2. Configuración de MercadoPago

1. **Crear cuenta de desarrollador**: https://www.mercadopago.com/developers
2. **Crear aplicación**
3. **Obtener credenciales**:
   - Public Key
   - Access Token
   - Client ID
   - Client Secret
4. **Configurar Webhooks**:
```
URL: https://tudominio.com/api/webhooks/mercadopago
   Eventos: payment, merchant_order
```

3. Configuración de AWS S3

1. **Crear bucket en S3**
2. **Configurar CORS**:
```
json
   [
     {
       "AllowedHeaders": ["*"],
       "AllowedMethods": ["GET", "PUT", "POST", "DELETE"],
       "AllowedOrigins": ["https://tudominio.com"],
       "ExposeHeaders": []
     }
   ]
```
3. **Crear usuario IAM** con permisos S3:
```
json
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Action": [
           "s3:GetObject",
           "s3:PutObject",
           "s3:DeleteObject"
         ],
         "Resource": "arn:aws:s3:::tu-bucket/*"
       }
     ]
   }
```

4. Configuración de Evolution API

1. **Configuración automática**: El contenedor se configura automáticamente
2. **Conectar WhatsApp**:
   - Acceder a: `https://tudominio.com/evolution/manager`
   - Crear nueva instancia
   - Escanear código QR con WhatsApp
   - Verificar conexión
3. **Webhooks automáticos**: Se configuran hacia `/api/webhooks/whatsapp`

📊 MONITOREO Y ALERTAS

1. Configuración de Portainer (Opcional)

```
# Habilitar Portainer
docker-compose --profile monitoring up -d portainer

# Acceder via: https://tudominio.com:9000
# Usuario: admin
# Crear contraseña en primer acceso
```

2. Script de Monitoreo Básico

```
#!/bin/bash
# Script: monitor.sh

EMAIL="admin@tudominio.com"
WEBHOOK_URL="https://hooks.slack.com/tu-webhook"  # Opcional

# Función para enviar alertas
send_alert() {
    local message="$1"
    echo "$message" | mail -s "ALERTA COMODÍN IA" $EMAIL

    # Opcional: Enviar a Slack
    curl -X POST -H 'Content-type: application/json' \
        --data "{\"text\":\"$message\"}" $WEBHOOK_URL
}

# Verificar servicios críticos
services=("app" "postgres" "redis" "evolution-api" "nginx")
for service in "${services[@]}"; do
    if ! docker-compose ps $service | grep -q "Up"; then
        send_alert "🚨 SERVICIO CAÍDO: $service en $(hostname)"
    fi
done

# Verificar uso de disco
disk_usage=$(df / | awk 'NR==2 {print $5}' | sed 's/%//')
if [ $disk_usage -gt 80 ]; then
    send_alert "🚨 DISCO LLENO: ${disk_usage}% en $(hostname)"
fi

# Verificar memoria
mem_usage=$(free | awk 'NR==2{printf "%.2f\n", $3*100/$2}')
if (( $(echo "$mem_usage > 90" | bc -l) )); then
    send_alert "🚨 MEMORIA ALTA: ${mem_usage}% en $(hostname)"
fi
```

3. Configurar Cron para Monitoreo

```
# Ejecutar monitoreo cada 5 minutos
crontab -e
# Agregar: */5 * * * * /ruta/a/monitor.sh
```

💾 RESPALDO Y RECUPERACIÓN

1. Configuración de Backups Automáticos

El sistema incluye backups automáticos de PostgreSQL. Para habilitar:

```
# Habilitar servicio de backup
docker-compose --profile backup up -d postgres-backup

# Los backups se crean en ./backups/ según esta programación:
# - Diarios: se mantienen 7 días
# - Semanales: se mantienen 4 semanas
# - Mensuales: se mantienen 6 meses
```

2. Backup Manual Completo

```
#!/bin/bash
# Script: backup-completo.sh

BACKUP_DIR="./backups/$(date +%Y%m%d_%H%M%S)"
mkdir -p "$BACKUP_DIR"

echo "=== Iniciando Backup Completo ==="

# 1. Base de datos
echo "Respaldando base de datos..."
docker-compose exec -T postgres pg_dump -U postgres comodin_ia | gzip > "$BACKUP_DIR/
database.sql.gz"

# 2. Redis
echo "Respaldando Redis..."
docker-compose exec -T redis redis-cli --rdb - | gzip > "$BACKUP_DIR/redis.rdb.gz"

# 3. Archivos de configuración
echo "Respaldando configuraciones..."
tar -czf "$BACKUP_DIR/config.tar.gz" .env docker-compose.yml nginx.conf ssl/

# 4. Volúmenes Docker
echo "Respaldando volúmenes..."
docker run --rm -v comodin_ia_postgres_data:/data -v "$PWD/backups:/backup" \
    ubuntu tar -czf "/backup/$(basename $BACKUP_DIR)/postgres_volume.tar.gz" -C /
data .

docker run --rm -v comodin_ia_evolution_instances:/data -v "$PWD/backups:/backup" \
    ubuntu tar -czf "/backup/$(basename $BACKUP_DIR)/evolution_volume.tar.gz" -C /
data .

# 5. Logs (últimos 7 días)
echo "Respaldando logs..."
find ./logs -name "*.log" -mtime -7 | tar -czf "$BACKUP_DIR/logs.tar.gz" -T -

echo "=== Backup Completo Finalizado ==="
echo "Ubicación: $BACKUP_DIR"
```

3. Procedimiento de Recuperación

```
#!/bin/bash
# Script: restaurar.sh
# Uso: ./restaurar.sh /ruta/a/backup/20240101_120000

BACKUP_DIR="$1"
if [ ! -d "$BACKUP_DIR" ]; then
    echo "Error: Directorio de backup no existe"
    exit 1
fi

echo "=== Iniciando Recuperación ==="
echo "Desde: $BACKUP_DIR"

# 1. Detener servicios
echo "Deteniendo servicios..."
docker-compose down

# 2. Restaurar configuraciones
echo "Restaurando configuraciones..."
tar -xzf "$BACKUP_DIR/config.tar.gz"

# 3. Restaurar volúmenes
echo "Restaurando volúmenes..."
docker volume create comodin_ia_postgres_data
docker run --rm -v comodin_ia_postgres_data:/data -v "$BACKUP_DIR:/backup" \
    ubuntu tar -xzf /backup/postgres_volume.tar.gz -C /data

# 4. Iniciar base de datos
echo "Iniciando base de datos..."
docker-compose up -d postgres redis
sleep 30

# 5. Restaurar base de datos
echo "Restaurando base de datos..."
gunzip -c "$BACKUP_DIR/database.sql.gz" | \
    docker-compose exec -T postgres psql -U postgres -d comodin_ia

# 6. Restaurar Redis
echo "Restaurando Redis..."
docker-compose stop redis
gunzip -c "$BACKUP_DIR/redis.rdb.gz" | \
    docker run --rm -i -v comodin_ia_redis_data:/data redis:7-alpine \
    sh -c 'cat > /data/dump.rdb'

# 7. Iniciar todos los servicios
echo "Iniciando todos los servicios..."
docker-compose up -d

echo "=== Recuperación Completa ==="
```

📞 CONTACTO Y SOPORTE

Información del Sistema

- **Versión**: 1.0.0
- **Última actualización**: $(date)

- **Documentación técnica**: Este archivo

Para Soporte Técnico

1. **Revisar logs**: `docker-compose logs [servicio]`
2. **Consultar troubleshooting**: Ver `TROUBLESHOOTING_GUIDE.md`
3. **Generar reporte del sistema**:
   `bash
   ./generar-reporte-sistema.sh > reporte-$(date +%Y%m%d).txt`

Información de Emergencia

- **Reinicio rápido**: `docker-compose restart`
- **Reinicio completo**: `docker-compose down && docker-compose up -d`
- **Backup de emergencia**: `./backup-completo.sh`

NOTA IMPORTANTE: Mantener este documento actualizado con cualquier cambio en la configuración o infraestructura. Este documento es tu guía definitiva para la gestión independiente de COMODÍN IA.
