



INSTRUCCIONES PARA EL AGENTE DEVOPS



RESUMEN DE LA MISIÓN

Realizar el **deployment completo** de la aplicación **COMODÍN IA** (CRM con WhatsApp) en el servidor VPS de producción.



OBJETIVO FINAL

- ✓ Aplicación funcionando en: `https://crm.comodinia.com`
- ✓ SSL configurado correctamente
- ✓ Base de datos Supabase conectada
- ✓ WhatsApp API (Evolution) funcionando
- ✓ Todos los servicios monitoreados y saludables



CREDENCIALES DEL SERVIDOR

IP: 31.97.175.147
Usuario: root
Contraseña: Komodin123456?
Dominio: crm.comodinia.com



UBICACIÓN DE LOS ARCHIVOS

La aplicación completa está en esta conversación en el directorio:

```
/home/ubuntu/comodin_ia/
```

ARCHIVOS CLAVE PREPARADOS:

- ✓ `docker-compose.production.yml` - Configuración Docker para producción
- ✓ `nginx.production.conf` - Configuración Nginx con SSL
- ✓ `scripts/deploy-production.sh` - Script automatizado principal
- ✓ `scripts/post-deploy.sh` - Script de finalización
- ✓ `scripts/verificar-deployment.sh` - Verificación completa
- ✓ `app/Dockerfile` - Container de la aplicación
- ✓ Todas las credenciales de producción configuradas

EJECUCIÓN PASO A PASO

PASO 1: Conectar al servidor

```
ssh root@31.97.175.147
# Contraseña: Komodin123456?
```

PASO 2: Limpiar ambiente anterior

```
# Eliminar configuración corrupta anterior
rm -f /etc/nginx/sites-enabled/*
rm -f /etc/nginx/sites-available/comodin*
rm -rf /srv/comodin_ia
systemctl reload nginx
```

PASO 3: Transferir aplicación

Copiar TODO el contenido de `/home/ubuntu/comodin_ia/` a `/srv/comodin_ia/` en el servidor.

PASO 4: Ejecutar deployment automatizado

```
cd /srv/comodin_ia
chmod +x scripts/*.sh
./scripts/deploy-production.sh
```

PASO 5: Finalizar deployment

```
./scripts/post-deploy.sh
```

PASO 6: Verificar resultado

```
./scripts/verificar-deployment.sh
```

CONFIGURACIONES CRÍTICAS

Base de Datos (Supabase)



- ☒ URL: `https://ovpcxvotqfmiqrdmloi.supabase.co`
- ☒ Todas las credenciales están configuradas
- ☒ Las migraciones se ejecutan automáticamente

SSL y Dominio

- ☒ Certificado Let's Encrypt configurado para `crm.comodinia.com`
- ☒ Redirección HTTP → HTTPS automática
- ☒ Headers de seguridad configurados

Servicios Docker

- ☒ **App principal:** Puerto 3000 (Next.js)

-  **Evolution API:** Puerto 8080 (WhatsApp)
-  **Nginx:** Puertos 80/443 (Proxy reverso)

CRITERIOS DE ÉXITO

1. Conectividad

- ☐ `https://crm.comodinia.com` responde
- ☐ Certificado SSL válido y activo
- ☐ Redirección HTTP → HTTPS funciona

2. Aplicación

- ☐ `/health` endpoint responde `200 OK`
- ☐ Login/registro funcional
- ☐ Base de datos conectada

3. WhatsApp API

- ☐ Evolution API accesible en puerto 8080
- ☐ Webhooks configurados correctamente

4. Monitoreo

- ☐ Logs de contenedores accesibles
- ☐ Métricas de sistema normales
- ☐ Backup automático configurado



SOLUCIÓN DE PROBLEMAS COMUNES

Error: “Aplicación no responde”

```
# Verificar contenedores
docker-compose -f docker-compose.production.yml ps
docker-compose -f docker-compose.production.yml logs app

# Reiniciar si es necesario
docker-compose -f docker-compose.production.yml restart
```

Error: “SSL Certificate Error”







```
# Regenerar certificado
certbot --nginx -d crm.comodinia.com --force-renewal
systemctl reload nginx
```

Error: “Database Connection”

```
# Verificar variables de entorno
docker-compose -f docker-compose.production.yml exec app env | grep DATABASE
```

ENTREGABLE FINAL

Una vez completado exitosamente, confirmar:

1.  **URL funcional:** <https://crm.comodinia.com>
2.  **Login page** carga correctamente
3.  **Certificado SSL** válido y configurado
4.  **Health check** responde: <https://crm.comodinia.com/health>
5.  **Evolution API** accesible para WhatsApp
6.  **Base de datos** conectada y migraciones aplicadas

CONFIRMACIÓN FINAL

Ejecutar el último comando de verificación y reportar:

```
cd /srv/comodin_ia
./scripts/verificar-deployment.sh
```

¡La aplicación COMODÍN IA debe estar 100% funcional en producción!

REFERENCIAS TÉCNICAS

- **Framework:** Next.js 14 con TypeScript
 - **Base de datos:** Supabase (PostgreSQL)
 - **Autenticación:** NextAuth.js
 - **Pagos:** Stripe + MercadoPago
 - **WhatsApp:** Evolution API
 - **Deployment:** Docker + Docker Compose
 - **Proxy:** Nginx con SSL
 - **Monitoreo:** Health checks automáticos
-

IMPORTANTE: Si encuentras errores durante el deployment, revisa los logs detalladamente y ejecuta los scripts de verificación para identificar la causa raíz.