



# MANUAL DE DEPLOYMENT - VIRA

---

Guía completa para hacer deploy de VIRA en diferentes plataformas de producción.



## OPCIONES DE DEPLOYMENT

---

### Recomendadas para Producción

- **Vercel** - Más fácil y optimizado para Next.js
- **Netlify** - Excelente para JAMstack
- **Railway** - Simple y con base de datos incluida
- **DigitalOcean App Platform** - Balance costo/performance

### Para Empresas Grandes

- **AWS ECS/Fargate** - Máxima escalabilidad
- **Google Cloud Run** - Serverless empresarial
- **Azure Container Instances** - Integración Microsoft
- **Kubernetes** - Control total



## DEPLOYMENT EN VERCEL (RECOMENDADO)

---

### Paso 1: Preparar el Proyecto

```
# Asegurar que el proyecto esté listo
yarn build
yarn start # Probar localmente

# Verificar archivos necesarios
ls -la .vercel/
cat vercel.json # Debe existir
```

## Paso 2: Configurar Vercel

```
// vercel.json
{
  "version": 2,
  "builds": [
    {
      "src": "package.json",
      "use": "@vercel/next"
    }
  ],
  "routes": [
    {
      "src": "/api/(.*)",
      "dest": "/api/$1"
    }
  ],
  "env": {
    "NODE_ENV": "production"
  },
  "functions": {
    "app/**/*.tsx": {
      "maxDuration": 30
    },
    "api/**/*.ts": {
      "maxDuration": 60
    }
  }
}
```

## Paso 3: Variables de Entorno en Vercel

```
# Instalar Vercel CLI
npm i -g vercel

# Login
vercel login

# Configurar variables de entorno
vercel env add NODE_ENV
# production

vercel env add NEXTAUTH_URL
# https://tu-dominio-en-vercel.vercel.app

vercel env add NEXTAUTH_SECRET
# tu_secret_aquí

# Continúa con todas las variables de .env...
```

## Paso 4: Deploy

```
# Deploy inicial
vercel

# Deploy de producción
vercel --prod

# Deploy con dominio personalizado
vercel --prod --domains tu-dominio.com
```

## Paso 5: Configurar Dominio Personalizado

1. En Vercel Dashboard → Settings → Domains
2. Agregar `tu-dominio.com`
3. Configurar DNS:

```
CNAME www tu-proyecto.vercel.app
A @ 76.76.19.61
```



## DEPLOYMENT EN NETLIFY

### Paso 1: Configurar netlify.toml

```
[build]
  command = "yarn build"
  publish = ".next"

[build.environment]
  NODE_ENV = "production"
  NEXT_TELEMETRY_DISABLED = "1"

[[plugins]]
  package = "@netlify/plugin-nextjs"

[functions]
  node_bundler = "esbuild"

[[redirects]]
  from = "/api/*"
  to = "/.netlify/functions/:splat"
  status = 200

[[headers]]
  for = "/*_next/static/*"
  [headers.values]
    Cache-Control = "public, max-age=31536000, immutable"
```

## Paso 2: Deploy con Netlify CLI

```
# Instalar CLI
npm i -g netlify-cli

# Login
netlify login

# Deploy
netlify deploy --build --prod
```



## DEPLOYMENT EN RAILWAY

### Paso 1: Crear railway.json

```
{
  "build": {
    "builder": "nixpacks",
    "buildCommand": "yarn build"
  },
  "deploy": {
    "startCommand": "yarn start",
    "healthcheckPath": "/api/health",
    "restartPolicyType": "ON_FAILURE"
  }
}
```

### Paso 2: Deploy

```
# Instalar Railway CLI
npm i -g @railway/cli

# Login
railway login

# Crear proyecto
railway new

# Deploy
railway up
```

### Paso 3: Configurar Base de Datos

```
# Agregar PostgreSQL
railway add --database postgresql

# Obtener URL de conexión
railway variables
# DATABASE_URL será generada automáticamente
```

## DEPLOYMENT EN AWS (AVANZADO)

### Opción A: AWS Amplify

#### Paso 1: Configurar amplify.yml

```
version: 1
applications:
  - appRoot: .
    frontend:
      phases:
        preBuild:
          commands:
            - yarn install
        build:
          commands:
            - yarn build
      artifacts:
        baseDirectory: .next
        files:
          - '**/*'
      cache:
        paths:
          - node_modules/**/*
```

#### Paso 2: Deploy con Amplify CLI

```
npm i -g @aws-amplify/cli
amplify configure
amplify init
amplify add hosting
amplify publish
```

### Opción B: ECS con Fargate

#### Paso 1: Crear Dockerfile

```
FROM node:18-alpine

WORKDIR /app

# Instalar dependencias
COPY package*.json yarn.lock ./
RUN yarn install --frozen-lockfile

# Copiar código fuente
COPY . .

# Build de la aplicación
RUN yarn build

# Exponer puerto
EXPOSE 3000

# Comando de inicio
CMD ["yarn", "start"]
```

#### Paso 2: Configurar docker-compose.yml

```

version: '3.8'
services:
  vira-app:
    build: .
    ports:
      - "3000:3000"
    environment:
      - NODE_ENV=production
      - DATABASE_URL=${DATABASE_URL}
      - NEXTAUTH_URL=${NEXTAUTH_URL}
    depends_on:
      - postgres

  postgres:
    image: postgres:14
    environment:
      POSTGRES_DB: vira
      POSTGRES_USER: vira_user
      POSTGRES_PASSWORD: ${DB_PASSWORD}
    volumes:
      - postgres_data:/var/lib/postgresql/data

volumes:
  postgres_data:

```

### Paso 3: Deploy con CDK/CloudFormation

```

// infrastructure/vira-stack.ts
import { Stack, StackProps } from 'aws-cdk-lib';
import * as ecs from 'aws-cdk-lib/aws-ecs';
import * as ec2 from 'aws-cdk-lib/aws-ec2';
import * as rds from 'aws-cdk-lib/aws-rds';

export class ViraStack extends Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, props);

    const vpc = new ec2.Vpc(this, 'ViraVPC');

    const cluster = new ecs.Cluster(this, 'ViraCluster', {
      vpc: vpc
    });

    const database = new rds.DatabaseInstance(this, 'ViraDB', {
      engine: rds.DatabaseInstanceEngine.postgres({
        version: rds.PostgresEngineVersion.VER_14
      }),
      vpc: vpc,
      credentials: rds.Credentials.fromGeneratedSecret('vira_user'),
      multiAz: true
    });

    // Continuar configuración...
  }
}

```

## DEPLOYMENT CON DOCKER

### Dockerfile de Producción

```
# Multi-stage build para optimización
FROM node:18-alpine AS deps
WORKDIR /app
COPY package*.json yarn.lock ./
RUN yarn install --frozen-lockfile --production

FROM node:18-alpine AS builder
WORKDIR /app
COPY package*.json yarn.lock ./
RUN yarn install --frozen-lockfile
COPY . .
RUN yarn build

FROM node:18-alpine AS runner
WORKDIR /app

RUN addgroup --system --gid 1001 nodejs
RUN adduser --system --uid 1001 nextjs

COPY --from=builder /app/public ./public
COPY --from=builder --chown=nextjs:nodejs /app/.next/standalone ./
COPY --from=builder --chown=nextjs:nodejs /app/.next/static ../next/static

USER nextjs
EXPOSE 3000
ENV PORT 3000

CMD ["node", "server.js"]
```

## Docker Compose para Desarrollo

```
version: '3.8'
services:
  app:
    build: .
    ports:
      - "3000:3000"
    environment:
      - NODE_ENV=production
    volumes:
      - ../.env:/app/.env:ro
    depends_on:
      - postgres
      - redis

  postgres:
    image: postgres:14-alpine
    environment:
      POSTGRES_DB: vira
      POSTGRES_USER: vira_user
      POSTGRES_PASSWORD: secure_password
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data

  redis:
    image: redis:7-alpine
    ports:
      - "6379:6379"
    volumes:
      - redis_data:/data

volumes:
  postgres_data:
  redis_data:
```

## CONFIGURACIÓN DE DOMINIO

### DNS Records Necesarios


```
# Para dominio principal
A @ 76.76.19.61 (IP del provider)
CNAME www tu-proyecto.vercel.app

# Para subdominios (opcional)
CNAME api api.tu-proyecto.vercel.app
CNAME admin admin.tu-proyecto.vercel.app




# Para email (opcional)
MX @ mail.tu-dominio.com (priority 10)
TXT @ "v=spf1 include:_spf.google.com ~all"
```

### SSL/HTTPS

La mayoría de proveedores lo manejan automáticamente:

-  Vercel - SSL automático



-  Netlify - SSL automático
-  Railway - SSL automático
-  AWS/GCP - Configurar manualmente

## OPTIMIZACIONES DE PRODUCCIÓN

---

### Configuración Next.js

```

// next.config.js
/** @type {import('next').NextConfig} */
const nextConfig = {
  // Optimizaciones de performance
  swcMinify: true,
  compress: true,
  poweredByHeader: false,

  // Optimizaciones de imágenes
  images: {
    domains: ['s3.amazonaws.com', 'tu-bucket.s3.amazonaws.com'],
    formats: ['image/webp', 'image/avif'],
    minimumCacheTTL: 3600
  },

  // Headers de seguridad
  async headers() {
    return [
      {
        source: '/(.*)',
        headers: [
          {
            key: 'X-Frame-Options',
            value: 'DENY'
          },
          {
            key: 'X-Content-Type-Options',
            value: 'nosniff'
          },
          {
            key: 'Referrer-Policy',
            value: 'strict-origin-when-cross-origin'
          }
        ]
      }
    ]
  },

  // Redirects
  async redirects() {
    return [
      {
        source: '/dashboard',
        destination: '/',
        permanent: true
      }
    ]
  },

  // Configuración experimental
  experimental: {
    serverActions: true,
    appDir: true
  }
}

module.exports = nextConfig

```

## Variables de Entorno Producción

```
# === CORE ===
NODE_ENV=production
NEXT_TELEMETRY_DISABLED=1
NEXTAUTH_URL=https://tu-dominio.com

# === OPTIMIZACIONES ===
NEXT_REVALIDATE=3600
NEXT_CACHE_MAX_AGE=86400

# === MONITORING ===
SENTRY_DSN=tu_sentry_dsn
GOOGLE_ANALYTICS_ID=GA-XXXXXXXXX

# === CDN ===
NEXT_PUBLIC_CDN_URL=https://cdn.tu-dominio.com
S3_CDN_URL=https://d123456789.cloudfront.net
```



## MONITOREO Y LOGGING

### Configurar Sentry

```
npm i @sentry/nextjs
```

```
// sentry.client.config.js
import * as Sentry from '@sentry/nextjs'

Sentry.init({
  dsn: process.env.SENTRY_DSN,
  environment: process.env.NODE_ENV,
  tracesSampleRate: 1.0,
  beforeSend(event) {
    if (event.exception) {
      const error = event.exception.values?.[0];
      if (error?.type === 'ChunkLoadError') {
        return null; // Ignorar errores de chunks
      }
    }
    return event;
  }
})
```

## Configurar Analytics

```
// lib/analytics.ts
declare global {
  interface Window {
    gtag: (...args: any[]) => void;
  }
}

export const trackEvent = (
  action: string,
  category: string,
  label?: string,
  value?: number
) => {
  if (typeof window !== 'undefined' && window.gtag) {
    window.gtag('event', action, {
      event_category: category,
      event_label: label,
      value: value
    });
  }
};

export const trackNewscastGeneration = (
  duration: number,
  cost: number,
  categories: string[]
) => {
  trackEvent('generate_newscast', 'user_action', categories.join(','), cost);
};
```

## CI/CD PIPELINES

### GitHub Actions

```
# .github/workflows/deploy.yml
name: Deploy to Production

on:
  push:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '18'
          cache: 'yarn'

      - run: yarn install --frozen-lockfile
      - run: yarn lint
      - run: yarn test
      - run: yarn build

  deploy:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: amondnet/vercel-action@v20
        with:
          vercel-token: ${ secrets.VERCEL_TOKEN }
          vercel-org-id: ${ secrets.VERCEL_ORG_ID }
          vercel-project-id: ${ secrets.VERCEL_PROJECT_ID }
          vercel-args: '--prod'
```

## GitLab CI/CD

```
# .gitlab-ci.yml
stages:
  - test
  - build
  - deploy

variables:
  NODE_VERSION: "18"

test:
  stage: test
  image: node:18-alpine
  script:
    - yarn install --frozen-lockfile
    - yarn lint
    - yarn test
    - yarn build

deploy_production:
  stage: deploy
  image: node:18-alpine
  script:
    - npm i -g vercel
    - vercel --token $VERCEL_TOKEN --prod --confirm
  only:
    - main
```

## HEALTH CHECKS Y MONITORING

### Health Check Endpoint

```
// pages/api/health.ts
import { NextApiRequest, NextApiResponse } from 'next';
import { supabase } from '@lib/supabase';

export default async function handler(
  req: NextApiRequest,
  res: NextApiResponse
) {
  try {
    // Verificar base de datos
    const { data, error } = await supabase
      .from('users')
      .select('count')
      .limit(1);

    if (error) throw error;

    // Verificar APIs externas (sample)
    const externalChecks = await Promise.allSettled([
      fetch(process.env.ELEVENLABS_BASE_URL + '/v1/user'),
      fetch(process.env.OPENWEATHER_API_URL)
    ]);

    const status = {
      status: 'healthy',
      timestamp: new Date().toISOString(),
      database: 'connected',
      external_apis: {
        elevenlabs: externalChecks[0].status === 'fulfilled' ? 'ok' : 'error',
        weather: externalChecks[1].status === 'fulfilled' ? 'ok' : 'error'
      },
      version: process.env.npm_package_version || '1.0.0'
    };

    res.status(200).json(status);
  } catch (error) {
    res.status(503).json({
      status: 'unhealthy',
      error: error.message,
      timestamp: new Date().toISOString()
    });
  }
}
```

### Uptime Monitoring

```
# Configurar con UptimeRobot, Pingdom, o similar
curl -X POST https://api.uptimerobot.com/v2/newMonitor \
  -d 'api_key=tu_api_key' \
  -d 'format=json' \
  -d 'type=1' \
  -d 'url=https://tu-dominio.com/api/health' \
  -d 'friendly_name=VIRA Health Check'
```



## SEGURIDAD EN PRODUCCIÓN

### Headers de Seguridad

```
// middleware.ts
import { NextResponse } from 'next/server';
import type { NextRequest } from 'next/server';

export function middleware(request: NextRequest) {
  const response = NextResponse.next();

  // Headers de seguridad
  response.headers.set('X-Frame-Options', 'DENY');
  response.headers.set('X-Content-Type-Options', 'nosniff');
  response.headers.set('Referrer-Policy', 'strict-origin-when-cross-origin');
  response.headers.set('X-XSS-Protection', '1; mode=block');

  // CSP Header
  response.headers.set(
    'Content-Security-Policy',
    "default-src 'self'; script-src 'self' 'unsafe-eval' 'unsafe-inline'; style-src 'self' 'unsafe-inline';"
  );

  return response;
}

export const config = {
  matcher: [
    '/((?!api|_next/static|_next/image|favicon.ico).*)',
  ],
};
```

## Rate Limiting

```
// lib/rate-limit.ts
import { NextApiRequest, NextApiResponse } from 'next';

const rateLimit = (limit: number, windowMs: number) => {
  const requests = new Map();

  return (req: NextApiRequest, res: NextApiResponse, next: () => void) => {
    const ip = req.headers['x-forwarded-for'] || req.socket.remoteAddress;
    const now = Date.now();
    const windowStart = now - windowMs;

    // Limpiar requests antiguos
    const userRequests = requests.get(ip) || [];
    const validRequests = userRequests.filter((time: number) => time > windowStart);

    if (validRequests.length >= limit) {
      return res.status(429).json({
        error: 'Rate limit exceeded',
        retryAfter: Math.ceil(windowMs / 1000)
      });
    }

    validRequests.push(now);
    requests.set(ip, validRequests);

    next();
  };
};

export default rateLimit;
```



## ESCALAMIENTO Y PERFORMANCE

### Database Optimization

```
-- Índices para performance
CREATE INDEX CONCURRENTLY idx_news_reports_user_status
ON news_reports(user_id, status);

CREATE INDEX CONCURRENTLY idx_scraped_news_category_date
ON scraped_news(category, scraped_at);

CREATE INDEX CONCURRENTLY idx_token_usage_user_date
ON token_usage(user_id, created_at);

-- Partitioning para tablas grandes
CREATE TABLE scraped_news_2024 PARTITION OF scraped_news
FOR VALUES FROM ('2024-01-01') TO ('2025-01-01');

-- Cleanup automático
CREATE OR REPLACE FUNCTION cleanup_old_data()
RETURNS void AS $$
BEGIN
    DELETE FROM scraped_news WHERE scraped_at < NOW() - INTERVAL '90 days';
    DELETE FROM system_logs WHERE timestamp < NOW() - INTERVAL '30 days';
END;
$$ LANGUAGE plpgsql;

-- Ejecutar cleanup diariamente
SELECT cron.schedule('cleanup-old-data', '0 2 * * *', 'SELECT cleanup_old_data();');
```

## Caching Strategy

```
// lib/cache.ts
import Redis from 'ioredis';

const redis = new Redis(process.env.REDIS_URL);

export const cache = {
  async get(key: string) {
    const value = await redis.get(key);
    return value ? JSON.parse(value) : null;
  },

  async set(key: string, value: any, ttl = 3600) {
    await redis.setex(key, ttl, JSON.stringify(value));
  },

  async invalidate(pattern: string) {
    const keys = await redis.keys(pattern);
    if (keys.length > 0) {
      await redis.del(...keys);
    }
  }
};

// Uso en APIs
export async function getCachedReports(userId: string) {
  const cacheKey = `user:${userId}:reports`;
  let reports = await cache.get(cacheKey);

  if (!reports) {
    reports = await getUserReports(userId);
    await cache.set(cacheKey, reports, 300); // 5 minutos
  }

  return reports;
}
```






## ✓ CHECKLIST DE DEPLOYMENT

### Pre-Deploy







- [ ] ☒ Tests pasan completamente
- [ ] ☒ Build de producción funciona
- [ ] ☒ Variables de entorno configuradas
- [ ] ☒ Base de datos migrada
- [ ] ☒ SSL/HTTPS configurado
- [ ] ☒ Dominio configurado correctamente
- [ ] ☒ Monitoring configurado
- [ ] ☒ Backups automáticos activos

### Post-Deploy

- [ ] ☒ Health check responde correctamente
- [ ] ☒ Login/logout funciona
- [ ] ☒ Generación de noticiero funciona

- [ ]  APIs externas responden
- [ ]  Archivos S3 se suben correctamente
- [ ]  Emails/notificaciones funcionan
- [ ]  Analytics reportando datos
- [ ]  Performance monitoring activo

## Monitoreo Continuo

- [ ]  Uptime monitoring configurado
- [ ]  Error tracking activo (Sentry)
- [ ]  Performance monitoring
- [ ]  Cost monitoring (AWS/GCP)
- [ ]  User analytics funcionando
- [ ]  Database performance monitoreada

---

## TROUBLESHOOTING DEPLOYMENT

---

### Build Errors

```
# Error: Module not found
rm -rf node_modules yarn.lock
yarn install

# Error: Out of memory
export NODE_OPTIONS="--max-old-space-size=4096"
yarn build

# Error: TypeScript errors
yarn tsc --noEmit
```

### Runtime Errors

```
# Verificar logs
vercel logs tu-proyecto
netlify logs
docker logs container_name

# Debug local
yarn build && yarn start
NODE_ENV=production yarn dev
```

## Database Issues

```
-- Verificar conexiones
SELECT * FROM pg_stat_activity WHERE state = 'active';

-- Verificar performance
SELECT query, mean_exec_time, calls FROM pg_stat_statements
ORDER BY mean_exec_time DESC LIMIT 10;

-- Reset conexiones
SELECT pg_terminate_backend(pid) FROM pg_stat_activity
WHERE state = 'idle in transaction' AND state_change < now() - INTERVAL '1 hour';
```

---

¡Tu aplicación VIRA estará lista para producción siguiendo esta guía! 🚀

Para soporte específico de deployment, consulta la documentación de cada proveedor o contacta al equipo técnico.