**1. Guía de Despliegue (deployment-guide.md)**

# Guía de Despliegue en Render

Esta guía describe los pasos para desplegar la aplicación "AI-Compliance Copilot" en Render utilizando el fichero `render.yaml`.

## Requisitos

1. \*\*Cuenta de Render\*\*: Necesitas una cuenta en [Render](<https://render.com/>).

2. \*\*Repositorio Git\*\*: El código fuente debe estar en un repositorio de GitHub o GitLab al que Render pueda acceder.

3. \*\*CLI de Render\*\* (Opcional): Para despliegues manuales o gestión desde la terminal, puedes instalar la [CLI de Render](<https://render.com/docs/cli>).

## Configuración del `render.yaml`

El fichero `render.yaml` en la raíz del proyecto define la infraestructura como código. Render lo utiliza para crear y configurar automáticamente los servicios.

El fichero actual define tres servicios principales:

1. `ai-compliance-copilot-api`: El backend de FastAPI.

2. `ai-compliance-copilot-web`: El frontend de Next.js.

3. `postgres`: La base de datos PostgreSQL.

4. `minio`: Un servicio de almacenamiento de objetos compatible con S3.

## Pasos para el Despliegue

1. \*\*Crear un "Blueprint Instance" en Render:\*\*

\* Ve a tu dashboard de Render.

\* Haz clic en \*\*"New"\*\* y selecciona \*\*"Blueprint"\*\*.

\* Conecta tu repositorio de Git donde se encuentra el proyecto.

\* Render detectará automáticamente el `render.yaml` y mostrará los servicios que se van a crear.

2. \*\*Configurar las Variables de Entorno y Secretos:\*\*

El `render.yaml` está configurado para usar "Secret Files" o grupos de variables de entorno para las credenciales sensibles. Antes de desplegar, asegúrate de haber creado un grupo de secretos en Render con las siguientes claves:

\* `AUTH0\_DOMAIN`

\* `AUTH0\_CLIENT\_ID`

\* `AUTH0\_CLIENT\_SECRET`

\* `AUTH0\_ISSUER`

\* `AUTH0\_AUDIENCE`

Render se encargará de inyectar estas variables en los servicios correspondientes durante el despliegue.

3. \*\*Revisar y Aprobar el Plan:\*\*

\* Render mostrará un resumen de los servicios y los costes asociados (los planes "free" no tienen coste).

\* Haz clic en \*\*"Create New Services"\*\* para iniciar el proceso de despliegue.

4. \*\*Proceso de Build y Despliegue:\*\*

\* Render clonará tu repositorio y, utilizando los Dockerfiles (`apps/api/Dockerfile` y `apps/web/Dockerfile`), construirás las imágenes de los servicios.

\* Una vez construidas, las desplegará y pondrá en marcha.

\* Puedes seguir el progreso de los logs de build y despliegue en tiempo real desde el dashboard de Render.

5. \*\*Verificar el Despliegue:\*\*

\* Una vez que los servicios estén marcados como "Live" o "Healthy", estarán listos.

\* La URL pública del frontend (`ai-compliance-copilot-web`) será la dirección principal de tu aplicación.

\* El backend (`ai-compliance-copilot-api`) será accesible a través de la URL que Render le asigne, y el frontend ya está configurado para comunicarse con ella.

## Actualizaciones y Despliegue Continuo

Por defecto, Render está configurado para el despliegue continuo. Cualquier `git push` a la rama `main` (o la rama que hayas configurado) disparará automáticamente un nuevo despliegue con los últimos cambios.

## Consideraciones Adicionales

\* \*\*Base de Datos\*\*: La base de datos PostgreSQL es persistente. Sus datos no se perderán entre despliegues.

\* \*\*Almacenamiento de Objetos\*\*: El servicio MinIO también tiene un disco persistente para que los archivos subidos no se pierdan.

\* \*\*Dominios Personalizados\*\*: Puedes configurar dominios personalizados para tus servicios web desde la configuración de cada servicio en el dashboard de Render.

---

**2. Archivo de Despliegue de Render (render.yaml)**

services:

- type: web

name: ai-compliance-copilot-api

runtime: docker

dockerfilePath: ./apps/api/Dockerfile

envVars:

- key: DATABASE\_URL

fromDatabase:

name: postgres

property: connectionString

- key: AUTH0\_DOMAIN

secretKey: AUTH0\_DOMAIN

- key: AUTH0\_AUDIENCE

secretKey: AUTH0\_AUDIENCE

- key: AUTH0\_ISSUER\_BASE\_URL

secretKey: AUTH0\_ISSUER

- key: S3\_ENDPOINT

fromService:

type: pserv

name: minio

property: host

- key: S3\_ACCESS\_KEY

fromService:

type: pserv

name: minio

envVarKey: MINIO\_ROOT\_USER

- key: S3\_SECRET\_KEY

fromService:

type: pserv

name: minio

envVarKey: MINIO\_ROOT\_PASSWORD

- key: S3\_BUCKET\_NAME

value: ai-compliance-evidence

- key: S3\_USE\_SSL

value: true

- key: NODE\_ENV

value: production

- key: PORT

value: 8000

- type: web

name: ai-compliance-copilot-web

runtime: docker

dockerfilePath: ./apps/web/Dockerfile

envVars:

- key: NEXT\_PUBLIC\_AUTH0\_DOMAIN

secretKey: AUTH0\_DOMAIN

- key: NEXT\_PUBLIC\_AUTH0\_CLIENT\_ID

secretKey: AUTH0\_CLIENT\_ID

- key: AUTH0\_CLIENT\_SECRET

secretKey: AUTH0\_CLIENT\_SECRET

- key: AUTH0\_SECRET

generateValue: true

- key: AUTH0\_BASE\_URL

value: [https://ai-compliance-copilot-web.onrender.com](https://ai-compliance-copilot-web.onrender.com/)

- key: AUTH0\_ISSUER\_BASE\_URL

secretKey: AUTH0\_ISSUER

- key: NEXT\_PUBLIC\_API\_URL

fromService:

type: web

name: ai-compliance-copilot-api

property: url

databases:

- name: postgres

databaseName: ai\_compliance\_copilot

user: postgres

privateServices:

- type: pserv

name: minio

env: docker

dockerImage: minio/minio

dockerCommand: server /data --console-address ":9001"

envVars:

- key: MINIO\_ROOT\_USER

generateValue: true

- key: MINIO\_ROOT\_PASSWORD

generateValue: true

### 3. Archivo de Despliegue de Fly.io (fly.toml)

# fly.toml app configuration file generated for ai-compliance-copilot

#

# See <https://fly.io/docs/reference/configuration/> for information about how to use this file.

#

app = "ai-compliance-copilot"

primary\_region = "mad"

[build]

dockerfile = "apps/api/Dockerfile"

[http\_service]

internal\_port = 8000

force\_https = true

auto\_stop\_machines = true

auto\_start\_machines = true

min\_machines\_running = 0

processes = ["app"]

[[vm]]

cpu\_kind = "shared"

cpus = 1

memory\_mb = 1024

---

### 4. Variables de Entorno (Ejemplos)

**API - apps/api/.env.example**

# Database

DATABASE\_URL=postgresql://user:password@localhost:5432/ai\_compliance\_copilot

# Auth0 Configuration

AUTH0\_DOMAIN=[your-auth0-domain.auth0.com](http://your-auth0-domain.auth0.com/)

AUTH0\_AUDIENCE=your-audience

AUTH0\_ISSUER\_BASE\_URL=<https://api.ai-compliance-copilot.com/>

# MinIO Configuration

MINIO\_ENDPOINT=localhost:9000

MINIO\_ACCESS\_KEY=minioadmin

MINIO\_SECRET\_KEY=minioadmin

MINIO\_BUCKET\_NAME=ai-compliance-evidence

# OpenAI Configuration (for RAG features)

OPENAI\_API\_KEY=your-openai-api-key

# Server Configuration

PORT=8000

NODE\_ENV=development

**Web - apps/web/.env.local.example**

# Auth0 Configuration

NEXT\_PUBLIC\_AUTH0\_DOMAIN=[your-auth0-domain.auth0.com](http://your-auth0-domain.auth0.com/)

NEXT\_PUBLIC\_AUTH0\_CLIENT\_ID=your-client-id

AUTH0\_CLIENT\_SECRET=your-client-secret

AUTH0\_SECRET=your-session-secret

AUTH0\_BASE\_URL=[http://localhost:3000](http://localhost:3000/)

AUTH0\_ISSUER\_BASE\_URL=<https://api.ai-compliance-copilot.com/>

# API Configuration

NEXT\_PUBLIC\_API\_URL=[http://localhost:8000](http://localhost:8000/)

### 5. README del Proyecto (README.md)

# AI-Compliance Copilot

MVP para cumplimiento de la Ley de IA de la UE y RGPD.

## Arquitectura

- \*\*apps/web\*\*: Frontend Next.js con Auth0

- \*\*apps/api\*\*: Backend Node.js con autenticación JWT

- \*\*apps/opa\*\*: Servicio Open Policy Agent para evaluación de riesgos

- \*\*PostgreSQL\*\*: Base de datos principal

- \*\*MinIO\*\*: Almacenamiento de evidencias (compatible con S3)

## Configuración rápida

1. \*\*Clonar y configurar dependencias:\*\*

```bash

git clone

cd ai-compliance-copilot

npm install

```

2. \*\*Configurar variables de entorno:\*\*

```bash

# En apps/api/

cp .env.example .env

# Editar .env con tus credenciales

# En apps/web/

cp .env.local.example .env.local

# Editar .env.local con tus credenciales Auth0

```

3. \*\*Levantar servicios con Docker:\*\*

```bash

docker-compose up -d postgres minio

```

4. \*\*Ejecutar migraciones:\*\*

```bash

cd apps/api

npm run migrate

```

5. \*\*Iniciar aplicación:\*\*

```bash

# Terminal 1: API

cd apps/api && npm run dev

# Terminal 2: Web

cd apps/web && npm run dev

# Terminal 3: OPA

cd apps/opa && npm run dev

```

## Funcionalidades Sprint 1

 \*\*Autenticación\*\*: Auth0 con roles admin/member

 \*\*CRUD Sistemas\*\*: Crear y gestionar sistemas de IA

 \*\*CRUD Datasets\*\*: Gestionar conjuntos de datos

 \*\*Evaluación de riesgos\*\*: Integración con OPA

 \*\*Generación DPIA\*\*: Documentos PDF automáticos

 \*\*Almacenamiento evidencias\*\*: MinIO con URLs pre-firmadas

 \*\*UI MVP\*\*: Wizard de 3 pasos

 \*\*CI/CD\*\*: GitHub Actions

## Despliegue

El proyecto está configurado para desplegarse en:

- \*\*API\*\*: Render/Fly.io

- \*\*Web\*\*: Vercel/Netlify

- \*\*Base de datos\*\*: PostgreSQL gestionado

- \*\*Almacenamiento\*\*: S3 o MinIO

## Estructura del proyecto

```

ai-compliance-copilot/

├── apps/

│ ├── web/ # Frontend Next.js

│ ├── api/ # Backend Node.js

│ └── opa/ # Open Policy Agent

├── infra/ # Configuración de infraestructura

└── docker-compose.yml

```

## Próximos sprints

- \*\*Sprint 2\*\*: Colaboración en tiempo real

- \*\*Sprint 3\*\*: RAG con documentos legales

- \*\*Sprint 4\*\*: Dashboards y reportes avanzados

---