# n8n: Automatización de flujos de trabajo

---

## Introducción

Este artículo documenta el despliegue de **n8n** como motor de automatización *self-hosted*, ejecutado mediante contenedores Docker, con persistencia externa y **ejecución desacoplada mediante task runners**.

n8n se integra en el stack como **capa transversal de orquestación**, actuando como punto de unión entre servicios internos (APIs, contenedores, scripts, webhooks) y servicios externos. El objetivo es centralizar lógica de automatización sin dispersarla en cada aplicación.

El enfoque prioriza **control, trazabilidad, aislamiento de ejecución y persistencia**, evitando dependencias SaaS y manteniendo el control completo sobre datos, credenciales y comportamiento interno.

---

## Enfoque general / Arquitectura

El despliegue se basa en una arquitectura desacoplada y preparada para carga real:

- **Instancia principal de n8n** actuando como plano de control:
  - UI web y API.
  - Gestión de workflows, credenciales y ejecuciones.
  - Orquestación y envío de tareas.
- **Task runners dedicados**:
  - Procesos independientes responsables de ejecutar workflows y nodos de código.
  - Aislados del plano de control para evitar bloqueos o degradación de la UI.
- **Base de datos PostgreSQL externa**:
  - Persistencia de workflows, ejecuciones y credenciales.
  - Eliminación de los límites de SQLite en concurrencia y fiabilidad.
- **Variables sensibles externalizadas** mediante `.env`.
- **Volúmenes persistentes** para estado interno y datos.
- Preparado para **exposición tras proxy inverso** (Caddy) si se requiere acceso HTTPS.

La separación entre control (n8n) y ejecución (task runners) permite escalar, aislar errores y asumir workflows con carga real sin comprometer la estabilidad del sistema.

---

# Requisitos previos

Se asumen resueltos previamente los siguientes puntos:

- Docker y Docker Compose operativos.
- Almacenamiento persistente disponible para volúmenes.
- Subdominio definido si se va a exponer externamente.
- Clave de cifrado generada previamente para proteger credenciales internas.

Estos requisitos condicionan decisiones clave, especialmente en seguridad, persistencia y aislamiento de ejecución.

---

# Desarrollo

## Qué se hizo y por qué

- **PostgreSQL externo**
  - Sustituye a SQLite para evitar problemas de concurrencia y corrupción.
  - Facilita reinicios, recreaciones del contenedor y migraciones futuras.
- **Ejecución desacoplada mediante task runners**
  - Los workflows y nodos de código no se ejecutan en la instancia principal.
  - La UI y la API permanecen responsivas incluso con flujos pesados o bloqueantes.
  - Permite escalar ejecución sin duplicar la interfaz web.
- **Variables sensibles en `.env`**
  - Separación clara entre configuración y credenciales.
  - Facilita rotación de secretos sin modificar manifests.
- **Cifrado interno activado**
  - Las credenciales nunca se almacenan en claro en la base de datos.
  - La estabilidad de la clave de cifrado es crítica para la integridad del sistema.
- **Opt-out explícito de telemetría**
  - Se desactiva el envío de diagnósticos para mantener control total sobre la instancia.
- **Diseño proxy-ready**
  - Preparado para funcionar tras HTTPS sin reconfiguraciones internas invasivas.

No se documentan comandos triviales ni pasos manuales: el despliegue se considera **reproducible directamente desde repositorio**.

---

## Configuración utilizada (solo enlaces)

Toda la configuración se mantiene versionada en Gitea:

- Repositorio de configuración de n8n

Contenido relevante:

- `.env`: credenciales, claves de cifrado y parámetros de entorno.
- `docker-compose.yml`: definición de servicios, runners, volúmenes, dependencias y red.

---

# Validación

Comprobaciones mínimas tras el despliegue:

- PostgreSQL inicia correctamente y supera su *healthcheck*.
- La instancia principal de n8n arranca sin errores de conexión.
- Los task runners se conectan al broker sin errores de autenticación.
- Los volúmenes persistentes permiten escritura sin problemas de permisos.
- La interfaz web responde correctamente tras reinicios.
- Las credenciales y workflows sobreviven a reinicios completos del stack.

La validación busca confirmar **funcionamiento estable**, no auditar exhaustivamente el sistema.

---

# Decisiones importantes o problemas detectados

- El *healthcheck* de PostgreSQL es crítico: sin él, n8n puede arrancar antes de que la base de datos esté lista y fallar de forma poco visible.
- La variable `N8N_ENCRYPTION_KEY` debe mantenerse estable; cambiarla invalida todas las credenciales existentes.
- Los task runners no realizan ninguna acción si no hay ejecuciones que requieran código o carga; su inactividad inicial es normal.
- Si se expone a Internet, es obligatorio definir correctamente `WEBHOOK_URL`, `N8N_HOST`, `N8N_PROTOCOL` y habilitar cookies seguras para evitar comportamientos inconsistentes.

---

# Resumen breve

n8n se despliega como servicio de automatización centralizado, persistente y **con ejecución desacoplada**, integrándose en el stack mediante Docker, PostgreSQL y task runners dedicados. El diseño prioriza estabilidad, seguridad, aislamiento y capacidad de crecimiento sin depender de servicios externos.

---

# Referencias

- Documentación de n8n
- Repositorio en Github de n8n

---

Revision #15
Created 2024-11-23 20:52:15 UTC by Juan Francisco
Updated 2026-01-05 18:12:37 UTC by Juan Francisco