Universidad de Almería

Máster en Ingeniería Informática

Integración de Tecnologías y Servicios Informáticos

Práctica 8

DevOps y GitOps: Despliegue y Versionado de Flujos de Trabajo

---

Autor: Johan Eduardo Cala Torra

Fecha: 10 de diciembre de 2025

Curso 2024-2025

Índice

1. Introducción

Esta práctica final cierra el ciclo de vida del desarrollo de software aplicado a la automatización con n8n. A lo largo de las siete prácticas anteriores, hemos construido, probado e integrado flujos de trabajo de forma manual en una única instancia de n8n. Este enfoque es ideal para el aprendizaje y el desarrollo, pero no es escalable ni robusto para un entorno de producción.

En esta práctica aplicamos los principios de **DevOps** y **GitOps** para:

- **Gestionar configuraciones** mediante variables de entorno
- **Versionar flujos de trabajo** con Git
- **Desacoplar secretos** de la lógica de negocio
- **Automatizar despliegues** con n8n-cli
- **Implementar CI/CD** con GitHub Actions

1.1. Objetivos de Aprendizaje

Al finalizar esta práctica, se habrán desarrollado las siguientes competencias:

1. **Gestión de Entornos:** Separar entornos de desarrollo y producción
2. **Gestión de Configuración:** Usar variables de entorno para credenciales y configuraciones
3. **Control de Versiones:** Integrar workflows en Git con historial de cambios
4. **Automatización de Despliegues:** Usar n8n-cli para despliegues programáticos
5. **Diseño de Flujos GitOps:** Implementar despliegue continuo desde Git

2. Fundamentos Teóricos

2.1. Variables de Entorno

Las variables de entorno permiten desacoplar la configuración de la lógica. En lugar de escribir valores directamente en los nodos (hardcoding), utilizamos expresiones que leen variables del entorno:

```
// Antes (hardcoded)
"documentId": "1Q8_sUQLC-0nMVQIkxBSgomoMJcaSkOi9QuDqTpzYZq8"

// Despu s (con variable de entorno)
"documentId": "={{ $env.SHEET_ID_CHISTES }}"
```

Ventajas:

- Cambio fácil entre entornos (dev/prod)
- Secretos no expuestos en el código
- Configuración centralizada
- Workflows genéricos y reutilizables

2.2. Control de Versiones con Git

Un flujo de trabajo de n8n es un archivo JSON que puede versionarse en Git. Esto proporciona:

- **Historial:** Registro completo de cambios
- **Diferencias:** Ver qué cambió entre versiones
- **Reversión:** Volver a versiones anteriores
- **Colaboración:** Trabajo en equipo con ramas

2.3. n8n-cli: Despliegue Automatizado

La CLI de n8n permite importar workflows de forma programática:

```
n8n import:workflow --input=workflow.json
```

Configuración mediante variables de entorno:

- `N8N_HOST`: URL de la instancia de n8n
- `N8N_API_KEY`: Clave de API para autenticación

2.4. GitOps

GitOps es una metodología donde Git es la única fuente de verdad para la infraestructura y las aplicaciones. Los cambios se realizan mediante pull requests y se despliegan automáticamente.

Principios aplicados:

1. **Declarativo:** Los workflows se definen en JSON
2. **Versionado:** Todo cambio queda registrado en Git
3. **Automatizado:** Los despliegues se ejecutan automáticamente
4. **Auditable:** Historial completo de quién cambió qué y cuándo

2.5. CI/CD con GitHub Actions

GitHub Actions permite automatizar el ciclo de vida del software. En esta práctica, implementamos un pipeline que:

1. Detecta cambios en workflows
2. Instala las dependencias necesarias
3. Despliega automáticamente a producción
4. Notifica el resultado del despliegue

3. Parte Guiada: Refactorización con Variables de Entorno

3.1. Objetivo

Refactorizar el flujo de trabajo de la Práctica 3 (clasificación de chistes con Google Sheets) para que utilice variables de entorno en lugar de valores hardcodeados.

3.2. Workflow Original

El workflow original de la Práctica 3 tenía el ID del documento de Google Sheets y los nombres de las hojas directamente en el código:

```
{
  "name": "Google Sheets",
  "type": "n8n-nodes-base.googleSheets",
  "parameters": {
    "documentId": "1Q8_sUQLC-0nMVQIkxBSgomoMJcaSkOi9QuDqTpzYZq8",
    "sheetName": "Programming"
  }
}
```

Problemas de este enfoque:

- Si cambia el documento, hay que modificar el workflow
- No se puede usar el mismo workflow en diferentes entornos
- El ID del documento queda expuesto en Git

3.3. Refactorización Realizada

Se modificó el workflow para usar variables de entorno:

```
{
  "name": "Google Sheets",
  "type": "n8n-nodes-base.googleSheets",
  "parameters": {
    "documentId": "={{ $env.SHEET_ID_CHISTES }}",
    "sheetName": "={{ $env.SHEET_NAME_PROG }}"
  }
}
```

3.4. Variables de Entorno Definidas

En el archivo `.env`:

```
# Google Sheets - Practica 3
SHEET_ID_CHISTES=1Q8_sUQLC-0nMVQIkxBSgomoMJcaSkOi9QuDqTpzYZq8
SHEET_NAME_PROG=Programming
SHEET_NAME_MISC=Misc
SHEET_NAME_DARK=Dark
```

3.5. Resultado

El workflow refactorizado se encuentra en:

`practica-08/workflows/p3-chistes-refactorizado.json`

Ventajas obtenidas:

- Workflow genérico y reutilizable
- Fácil cambio de documento sin modificar el workflow
- Configuración centralizada en `.env`
- Mismo workflow funciona en dev y prod con diferentes `.env`

4. Ejercicio 1: Refactorización PostgreSQL + RabbitMQ

4.1. Objetivo

Refactorizar el flujo de trabajo de la Práctica 6 (sistema de microservicios con PostgreSQL y RabbitMQ) para usar variables de entorno en las credenciales.

4.2. Credenciales Refactorizadas

4.2.1. PostgreSQL

Se creó una nueva credencial que usa variables de entorno:

```
{
  "name": "PostgreSQL (Refactorizado)",
  "type": "postgres",
  "data": {
    "host": "={{ $env.DB_HOST }}",
    "port": "={{ $env.DB_PORT }}",
    "database": "={{ $env.DB_NAME }}",
    "user": "={{ $env.DB_USER }}",
    "password": "={{ $env.DB_PASS }}"
  }
}
```

4.2.2. RabbitMQ

De forma similar, se refactorizó la credencial de RabbitMQ:

```
{
  "name": "RabbitMQ (Refactorizado)",
  "type": "rabbitmq",
  "data": {
    "hostname": "={{ $env.RABBITMQ_HOST }}",
    "port": "={{ $env.RABBITMQ_PORT }}",
```

```
    "username": "={{ $env.RABBITMQ_USER }}",
    "password": "={{ $env.RABBITMQ_PASS }}"
  }
}
```

4.3. Variables de Entorno

En el archivo `.env`:

```
# PostgreSQL - Practica 6
DB_HOST=localhost
DB_PORT=5433
DB_NAME=taskdb
DB_USER=user
DB_PASS=password

# RabbitMQ - Practica 6
RABBITMQ_HOST=localhost
RABBITMQ_PORT=5672
RABBITMQ_USER=guest
RABBITMQ_PASS=guest
RABBITMQ_QUEUE_NAME=task_created
```

4.4. Workflow Refactorizado

El workflow completo se encuentra en:

`practica-08/workflows/p6-postgres-rabbitmq-refactorizado.json`

Este workflow demuestra cómo usar las credenciales refactorizadas en nodos de PostgreSQL y RabbitMQ.

5. Ejercicio 2: Docker Compose con Variables de Entorno

5.1. Objetivo

Migrar de `docker run` a `docker-compose.yml` con soporte completo para variables de entorno mediante archivo `.env`.

5.2. Archivo docker-compose.yml

Se creó un archivo `docker-compose.yml` que:

- Carga variables desde `.env`
- Pasa las variables al contenedor de n8n
- Se conecta a la red del sistema de microservicios

- Configura autenticación básica

```yaml
version: '3.8'

services:
  n8n:
    image: n8nio/n8n:latest
    container_name: n8n-practica-08
    restart: unless-stopped
    ports:
      - "5678:5678"

    env_file:
      - ./.env

    environment:
      # Autenticacion basica
      - N8N_BASIC_AUTH_ACTIVE=${N8N_BASIC_AUTH_ACTIVE}
      - N8N_BASIC_AUTH_USER=${N8N_BASIC_AUTH_USER}
      - N8N_BASIC_AUTH_PASSWORD=${N8N_BASIC_AUTH_PASSWORD}

      # Google Sheets
      - SHEET_ID_CHISTES=${SHEET_ID_CHISTES}
      - SHEET_NAME_PROG=${SHEET_NAME_PROG}
      - SHEET_NAME_MISC=${SHEET_NAME_MISC}
      - SHEET_NAME_DARK=${SHEET_NAME_DARK}

      # PostgreSQL
      - DB_HOST=${DB_HOST}
      - DB_PORT=${DB_PORT}
      - DB_NAME=${DB_NAME}
      - DB_USER=${DB_USER}
      - DB_PASS=${DB_PASS}

      # RabbitMQ
      - RABBITMQ_HOST=${RABBITMQ_HOST}
      - RABBITMQ_PORT=${RABBITMQ_PORT}
      - RABBITMQ_USER=${RABBITMQ_USER}
      - RABBITMQ_PASS=${RABBITMQ_PASS}
      - RABBITMQ_QUEUE_NAME=${RABBITMQ_QUEUE_NAME}

    volumes:
      - ./n8n-data:/home/node/.n8n

    networks:
      - n8n-network
      - task-network

networks:
  n8n-network:
    driver: bridge
```

```
  task-network:
    external: true
    name: task-manager-system_default
```

5.3. Archivo .env

El archivo `.env` contiene todas las variables de configuración:

```
# ==========================================
# Configuracion de n8n - Practica 08
# ==========================================

# Autenticacion de n8n
N8N_BASIC_AUTH_ACTIVE=true
N8N_BASIC_AUTH_USER=admin
N8N_BASIC_AUTH_PASSWORD=admin123

# Google Sheets - Practica 3
SHEET_ID_CHISTES=1Q8_sUQLC-0nMVQIkxBSgomoMJcaSkOi9QuDqTpzYZq8
SHEET_NAME_PROG=Programming
SHEET_NAME_MISC=Misc
SHEET_NAME_DARK=Dark

# PostgreSQL - Practica 6
DB_HOST=localhost
DB_PORT=5433
DB_NAME=taskdb
DB_USER=user
DB_PASS=password

# RabbitMQ - Practica 6
RABBITMQ_HOST=localhost
RABBITMQ_PORT=5672
RABBITMQ_USER=guest
RABBITMQ_PASS=guest
RABBITMQ_QUEUE_NAME=task_created

# n8n CLI (para despliegue automatizado)
N8N_API_KEY=
```

5.4. Archivo .env.example

Para compartir la estructura sin exponer secretos, se creó `.env.example`:

```
# Template de variables de entorno
# Copia este archivo a .env y configura tus valores

N8N_BASIC_AUTH_ACTIVE=true
N8N_BASIC_AUTH_USER=admin
N8N_BASIC_AUTH_PASSWORD=tu-password-aqui
```

```
SHEET_ID_CHISTES=tu-sheet-id-aqui
SHEET_NAME_PROG=Programming
# ... etc
```

5.5. Uso

Para iniciar n8n con las variables de entorno:

```
# Copiar template
cp .env.example .env

# Editar .env con valores reales
# ...

# Iniciar n8n
docker-compose up -d
```

6. Ejercicio 3: Pipeline CI/CD con GitHub Actions

6.1. Objetivo

Implementar un pipeline de integración y despliegue continuo que automáticamente despliegue los workflows a producción cuando se detecten cambios en Git.

6.2. Archivo de Workflow

Se creó el archivo `.github/workflows/deploy.yml`:

```
name: Deploy n8n Workflows

on:
  push:
    branches:
      - main
    paths:
      - 'practica-08/workflows/*.json'

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout repository
        uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18'

```

```
      - name: Install n8n CLI
        run: npm install n8n -g

      - name: Deploy workflows to n8n
        env:
          N8N_HOST: ${{ secrets.N8N_PROD_HOST }}
          N8N_API_KEY: ${{ secrets.N8N_PROD_API_KEY }}
        run: |
          for file in practica-08/workflows/*.json; do
            echo "Deploying $file..."
            n8n import:workflow --input="$file"
          done

      - name: Deployment summary
        run: |
          echo "Deployment completed successfully"
          echo "Workflows deployed from practica-08/workflows/"
```

6.3. Configuración de Secrets

En GitHub, se deben configurar los siguientes secrets:

- `N8N_PROD_HOST`: URL de la instancia de n8n en producción
- `N8N_PROD_API_KEY`: API Key generada en n8n

Pasos para configurar:

1. Ir a `Settings >Secrets and variables >Actions`
2. Click en `New repository secret`
3. Añadir cada secret con su valor correspondiente

6.4. Funcionamiento del Pipeline

1. **Trigger:** Se ejecuta al hacer push a `main`
2. **Filtro:** Solo si cambian archivos en `workflows/*.json`
3. **Checkout:** Descarga el código del repositorio
4. **Setup:** Instala Node.js 18 y n8n-cli
5. **Deploy:** Itera sobre todos los workflows y los despliega
6. **Summary:** Muestra un resumen del despliegue

6.5. Ventajas del Pipeline

- **Automatización completa:** No requiere intervención manual
- **Despliegue rápido:** Cambios en producción en minutos
- **Auditoría:** Historial completo en GitHub Actions
- **Rollback fácil:** Revertir commit = revertir despliegue
- **Notificaciones:** GitHub notifica éxito o fallo

7. Scripts de Despliegue Automatizado

Para facilitar el despliegue local, se crearon scripts para Windows y Linux/Mac.

7.1. Script para Windows (PowerShell)

El archivo `deploy.ps1` automatiza el proceso de despliegue:

```
# Verificar que existe .env
if (-Not (Test-Path .env)) {
    Write-Error "No se encontro el archivo .env"
    exit 1
}

# Iniciar n8n si no esta corriendo
$n8nRunning = docker ps --filter "name=n8n-practica-08"
if (-Not $n8nRunning) {
    docker-compose up -d
}

# Instalar n8n-cli si no esta instalado
try {
    Get-Command n8n -ErrorAction Stop
} catch {
    npm install n8n -g
}

# Desplegar workflows
$workflows = Get-ChildItem -Path .\workflows -Filter "*.json"
foreach ($workflow in $workflows) {
    n8n import:workflow --input="$($workflow.FullName)"
}
```

7.2. Script para Linux/Mac (Bash)

El archivo `deploy.sh` proporciona la misma funcionalidad:

```
#!/bin/bash
set -e
```

```
# Verificar .env
if [ ! -f .env ]; then
    echo "Error: No se encontro .env"
    exit 1
fi

# Iniciar n8n
if ! docker ps | grep -q "n8n-practica-08"; then
    docker-compose up -d
    sleep 10
fi

# Instalar n8n-cli
if ! command -v n8n &> /dev/null; then
    npm install n8n -g
fi

# Desplegar workflows
for workflow in ./workflows/*.json; do
    echo "Desplegando: $workflow"
    n8n import:workflow --input="$workflow"
done
```

7.3. Uso de los Scripts

En Windows:

```
cd practica-08
.\deploy.ps1
```

En Linux/Mac:

```
cd practica-08
chmod +x deploy.sh
./deploy.sh
```

8. Pruebas y Verificación

8.1. Verificación de Variables de Entorno

Para verificar que las variables de entorno están correctamente cargadas en n8n:

1. Crear un nodo **Set** temporal
2. Usar la expresión: {{ `$env.DB_HOST` }}
3. Ejecutar el nodo
4. Verificar que se muestra el valor: `localhost`

Si el valor se muestra correctamente, las variables están bien configuradas.

8.2. Prueba del Workflow de Google Sheets

1. Importar `p3-chistes-refactorizado.json`
2. Verificar que la credencial de Google Sheets está configurada
3. Ejecutar el workflow manualmente
4. Verificar que los chistes se guardan en Google Sheets
5. Comprobar que no hay errores de "variable not found"

8.3. Prueba del Workflow de PostgreSQL + RabbitMQ

Prerequisitos:

- Sistema de microservicios de Práctica 5 corriendo
- PostgreSQL accesible en puerto 5433
- RabbitMQ accesible en puerto 5672

Pasos:

1. Crear las credenciales refactorizadas en n8n
2. Actualizar los IDs de credenciales en el workflow JSON
3. Importar `p6-postgres-rabbitmq-refactorizado.json`
4. Ejecutar el workflow
5. Verificar conexión a PostgreSQL
6. Verificar envío de mensajes a RabbitMQ

8.4. Prueba del Despliegue Automatizado

1. Ejecutar el script de despliegue (`deploy.ps1` o `deploy.sh`)
2. Verificar que n8n inicia correctamente
3. Verificar que n8n-cli se instala
4. Verificar que los workflows se importan
5. Acceder a http://localhost:5678
6. Comprobar que los workflows están disponibles

8.5. Prueba del Pipeline CI/CD

Nota: Esta prueba requiere configurar GitHub Actions en un repositorio real.

1. Configurar secrets en GitHub
2. Hacer un cambio en un workflow
3. Commitear y pushear a `main`
4. Ir a `Actions` en GitHub
5. Verificar que el pipeline se ejecuta
6. Comprobar que el despliegue es exitoso
7. Verificar en n8n producción que el workflow se actualizó

9. Conclusiones

9.1. Objetivos Alcanzados

Esta práctica ha permitido aplicar principios profesionales de DevOps y GitOps a la automatización con n8n:

- **Gestión de Configuración:** Separación completa entre código y configuración mediante variables de entorno
- **Control de Versiones:** Workflows versionados en Git con historial completo de cambios
- **Seguridad:** Secretos externalizados y no expuestos en el código
- **Automatización:** Despliegues automatizados con n8n-cli y GitHub Actions
- **Reproducibilidad:** Entornos reproducibles mediante Docker Compose y `.env`

9.2. Competencias Desarrolladas

1. **DevOps:** Aplicación de prácticas de desarrollo y operaciones
2. **GitOps:** Git como fuente de verdad para infraestructura
3. **CI/CD:** Diseño e implementación de pipelines de despliegue continuo
4. **Infraestructura como Código:** Definición declarativa de entornos
5. **Gestión de Secretos:** Manejo seguro de credenciales y configuraciones

9.3. Aplicabilidad Profesional

Los conocimientos adquiridos en esta práctica son directamente aplicables en entornos profesionales:

- **Startups:** Despliegue rápido de automatizaciones
- **Empresas:** Gestión de múltiples entornos (dev, staging, prod)
- **Equipos:** Colaboración mediante Git y pull requests
- **Auditoría:** Trazabilidad completa de cambios
- **Escalabilidad:** Fácil replicación de entornos

9.4. Evolución del Aprendizaje

A lo largo de las 8 prácticas, hemos recorrido el ciclo completo de desarrollo:

1. **P1-P2:** Fundamentos de n8n y lógica de workflows
2. **P3-P4:** Procesamiento de datos y manejo de errores
3. **P5-P6:** Arquitectura de microservicios y mensajería
4. **P7:** Integración con servicios de IA
5. **P8:** DevOps, GitOps y producción

Este recorrido refleja el proceso real de desarrollo de software: desde el aprendizaje básico hasta el despliegue en producción con prácticas profesionales.

9.5. Reflexión Final

La práctica demuestra que n8n no es solo una herramienta de automatización, sino una plataforma completa que puede integrarse en flujos de trabajo profesionales con:

- Control de versiones
- Gestión de configuración
- Despliegue automatizado
- Integración continua
- Entrega continua

Estos principios son fundamentales en el desarrollo de software moderno y aplicables a cualquier tecnología o plataforma.