

---

**Realizado por:**

**Daniela Marin**

**Marco Antonio**

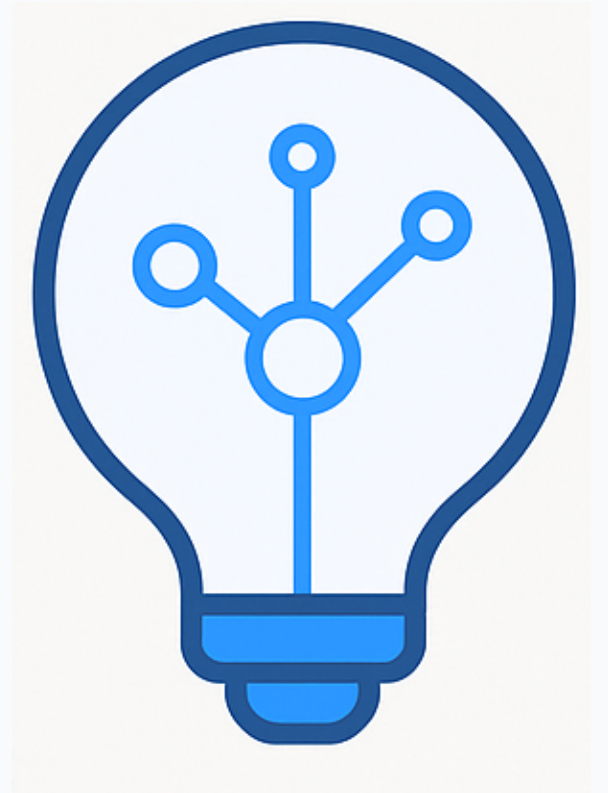
**Aragon**

**Laura Astudillo**

**Maria Paula Pinillo**

**Dillan Molina**

# Proyecto Redes e Infraestructura **TASKFLOW**



**Docente:**

**Oscar Mondragon**

---

# Índice de **CONTENIDOS**



---

01. Introducción

---

---

02. Datos y Dataset

---

---

03. Infraestructura y Arquitectura

---

---

04. Flujo y Procesamiento de Datos

---

---

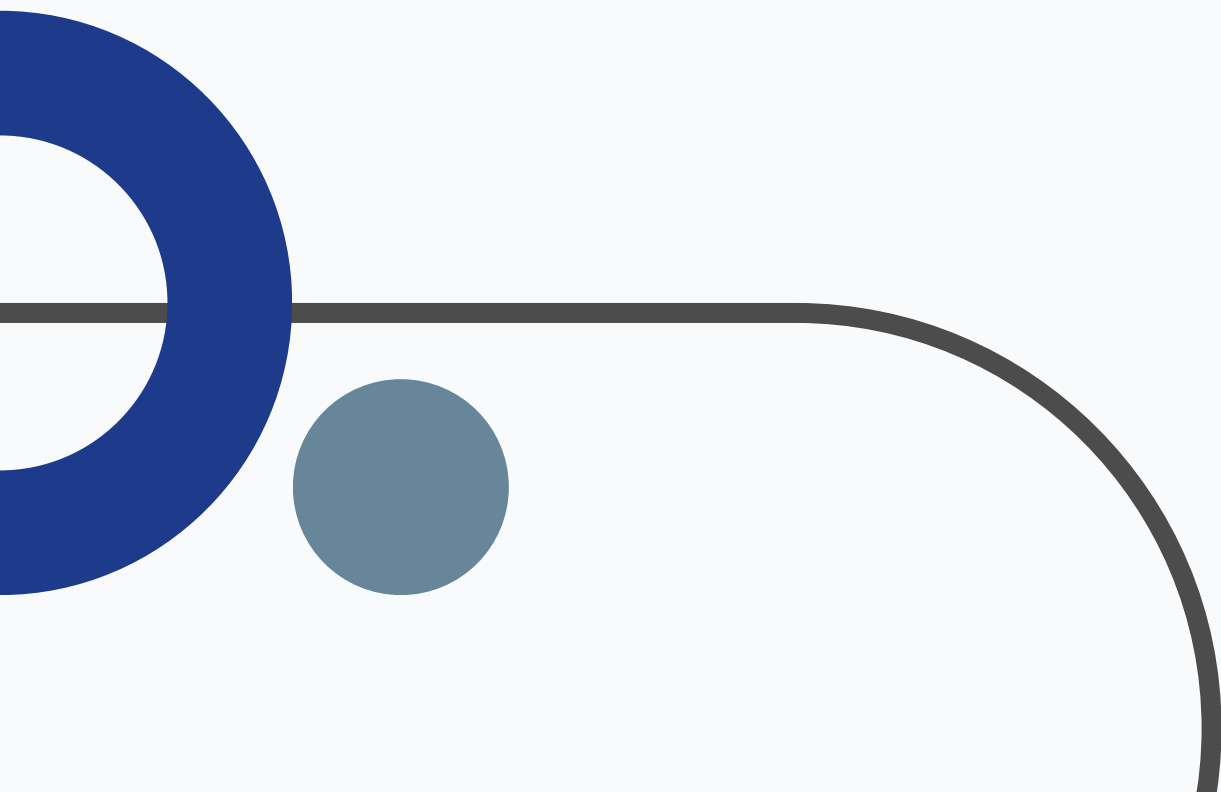
05. Implementación y Despliegue

---

---

05. Contacto

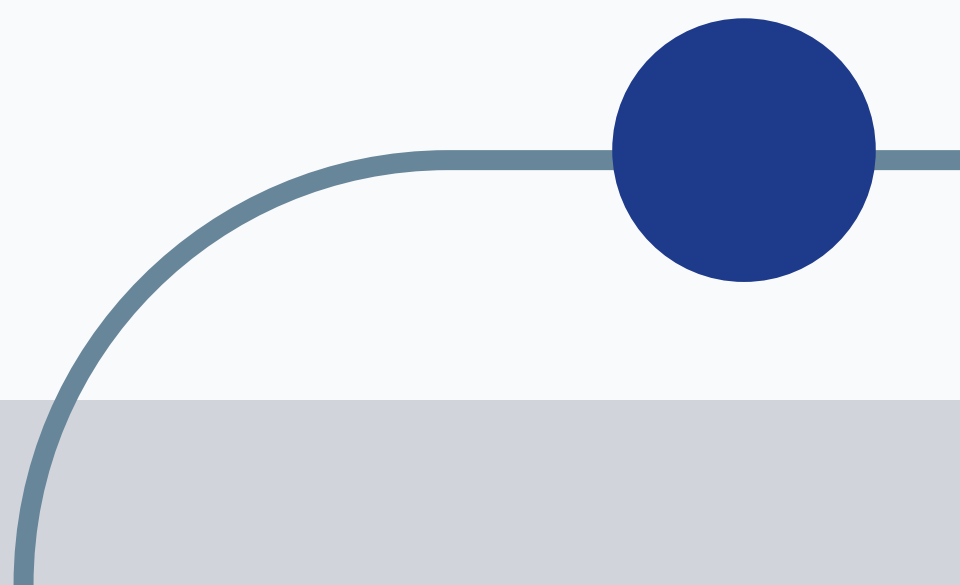
---



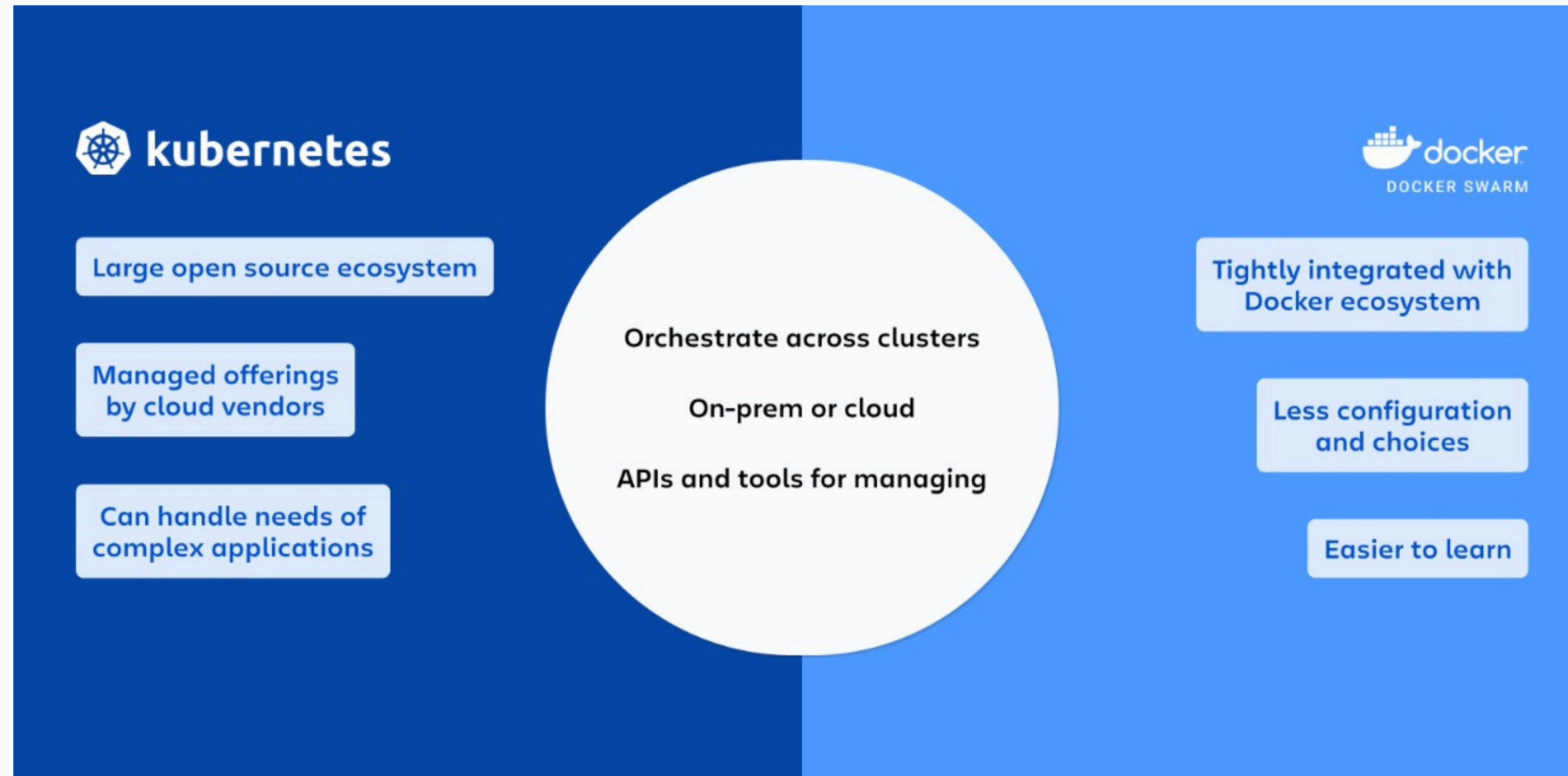


# ➔ Introducción ➔

Como proyecto para la asignatura de Redes e Infraestructura, decidimos desarrollar una aplicación web orientada a la gestión de empleados y asignación de tareas dentro de una empresa. Esta solución se complementa con un módulo de análisis que presenta visualizaciones gráficas sobre las actividades realizadas por los empleados.



# Análisis de alternativas





# DATOS Y DATASET

El dataset fue generado bajo el nombre df\_company y posteriormente cargado en una base de datos MySQL a través del script init.sql

Los principales campos incluidos son:

<b>project_id</b>	<b>id</b>	<b>Calidad</b>
<b>project_name</b>	<b>empleado_id</b>	<b>Iniciativa</b>
<b>boss_id</b>	<b>task_id</b>	<b>comunicacion</b>
<b>boss_name</b>	<b>fecha asignacion</b>	<b>satisfaccion_cliente</b>
<b>status: (Created, In Progress, Completed).</b>	<b>fecha_entrega_ maxima</b>	<b>calificacion_promedio</b>
<b>team_members</b>	<b>fecha_entrega</b>	



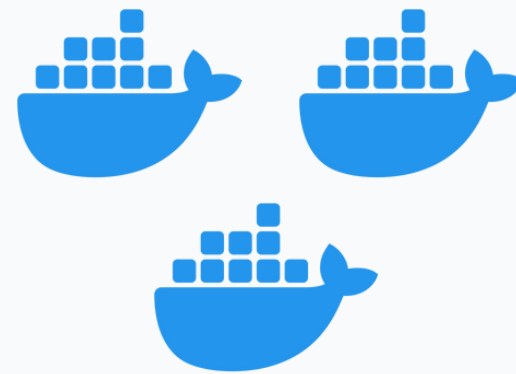
# ➔ INFRAESTRUCTURA Y ARQUITECTURA

CONTENERIZACIÓN



DOCKER

ORQUESTACIÓN



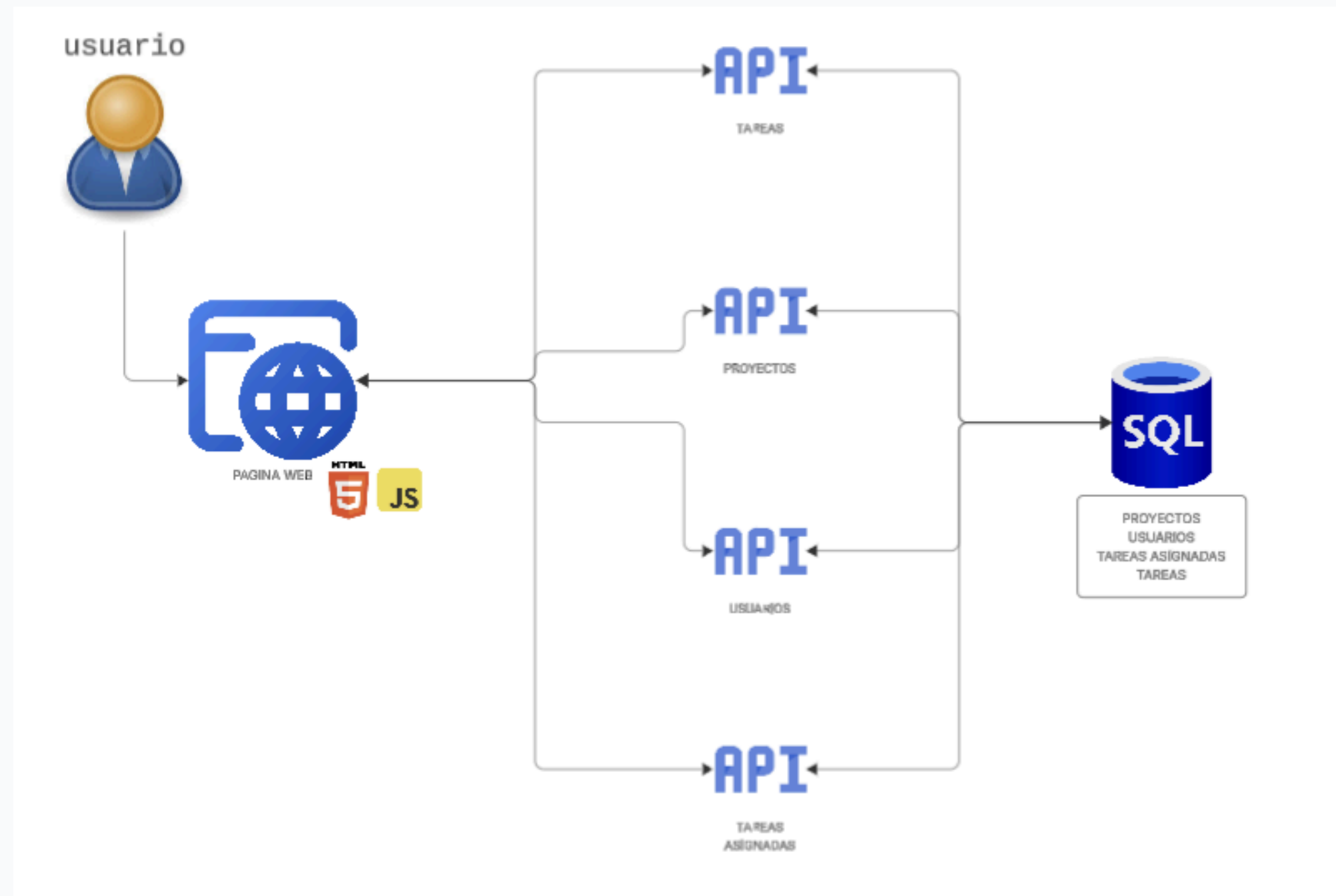
DOCKER SWARM

PROCESAMIENTO DE DATOS



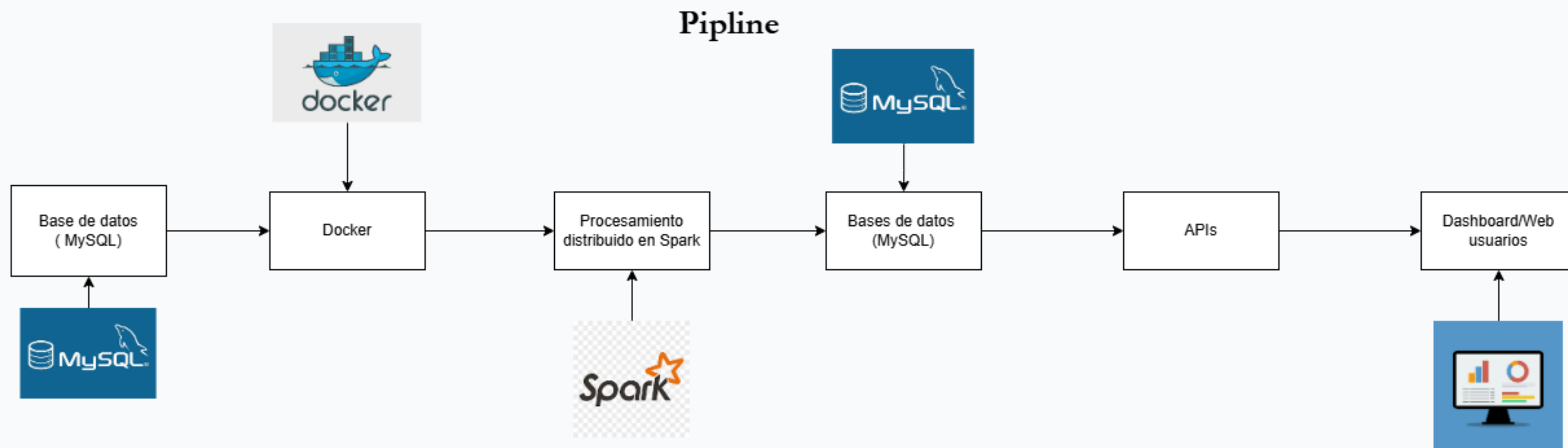
APACHE SPARK

# ➔ INFRAESTRUCTURA Y ARQUITECTURA



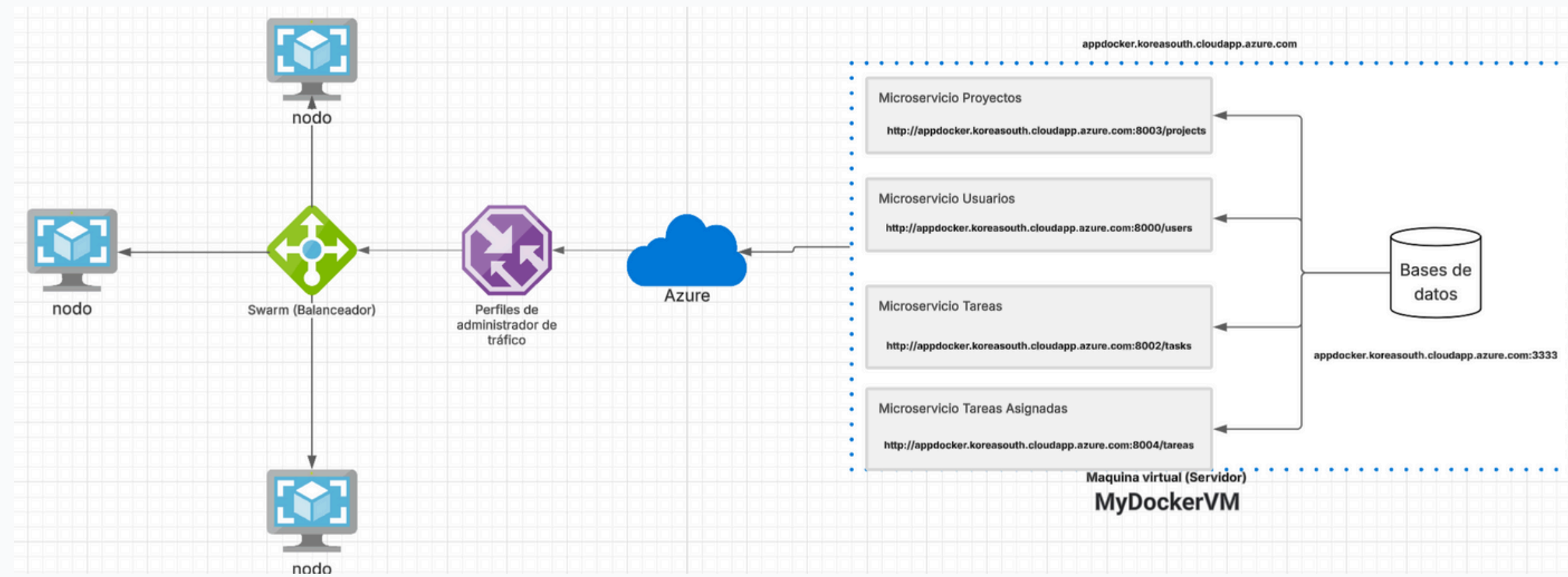
# ➔ Flujo y Procesamiento de Datos

## Diagrama pipeline





# Diagrama de implementación



# PRUEBAS DE CARGA

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/...	Sent KB/sec
HTTP Request	118296	1585	1592	1702	1754	5026	7	17517	0.00%	1462.9/sec	22362.09	174.29
TOTAL	118296	1585	1592	1702	1754	5026	7	17517	0.00%	1462.9/sec	22362.09	174.29

SettingsGraph

En el servicio [http:// appdocker.koreasouth.cloudapp.azure.com:8000/users](http://appdocker.koreasouth.cloudapp.azure.com:8000/users), también con una sola réplica, se alcanzaron más de 100,000 solicitudes exitosas sin errores.

# PRUEBAS DE ESCALABILIDAD

Filename	Browse... Log/Display Only: <input type="checkbox"/> Errors <input type="checkbox"/> Successes <input type="button" value="Configure"/>											
Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/...	Sent KB/sec
HTTP Request	57	46437	40292	89950	95938	99940	299	101924	100.00%	33.5/min	1.80	0.01
TOTAL	57	46437	40292	89950	95938	99940	299	101924	100.00%	33.5/min	1.80	0.01

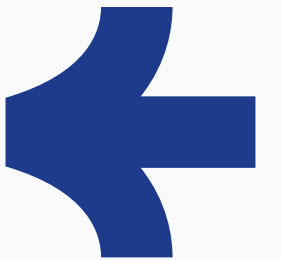
Con 5 réplicas y 50 solicitudes en 200 segundos, el sistema presentó más del 90 % de fallos, lo que sugiere que no todos los nodos respondieron adecuadamente.

# IMPLEMENTACIÓN Y DESPLIEGUE

## Implementación de la Solución

Cada componente del sistema fue encapsulado en su propio contenedor Docker mediante la creación de archivos Dockerfile. Estos archivos especificaron las dependencias necesarias, los comandos de instalación y la configuración para ejecutar los servicios en ambientes aislados.

Para facilitar el despliegue conjunto, se utilizó un archivo docker-compose.yml que define todos los servicios, redes, volúmenes y variables de entorno requeridas. Este archivo también sirve como base para el despliegue en Docker Swarm.



# CONCLUSIONES

- Se implementó una arquitectura contenerizada basada en microservicios con Docker y Docker Swarm.
- El sistema integra procesamiento distribuido con PySpark y visualización web dinámica.
- El dataset propio permitió un análisis realista y alineado con el objetivo del sistema.
- Las pruebas demostraron que la solución es funcional, escalable y modular.
- La arquitectura respondió de forma estable, incluso bajo carga, y automatiza todo el flujo de datos.





# Referencias

- [1] S. Newman, *\*Building Microservices: Designing Fine-Grained Systems\**, 1st ed., Sebastopol, CA: O'Reilly Media, 2015. [DMV1]
- [2] [2] D. Merkel, "Docker: Lightweight Linux Containers for Consistent Development and Deployment," *\*Linux Journal\**, vol. 2014, no. 239, pp. 2–11, Mar. 2014.
- [3] M. Zaharia, B. Chambers, and M. Xin, *\*Spark: The Definitive Guide\**, 1st ed. Sebastopol, CA: O'Reilly Media, 2018.
- [4] D. Merkel, "Docker: Lightweight Linux containers for consistent development and deployment," *Linux Journal*, vol. 2014, no. 239, pp. 2–2, 2014.
- [5] Docker Inc., "Swarm mode overview," Docker Documentation. [Online]. Available: <https://docs.docker.com/engine/swarm/> [Accessed: May 22, 2025].
- [6] Apache\_Spark. Accessed: Aug. 12, 2024. [Online]. Available: [Diagrama pipeline](#)  
.
- [7] Microsoft, "Introducción para desarrolladores de Azure," Microsoft Learn, 2023. [En línea]. Disponible: <https://learn.microsoft.com/es-es/azure/developer/intro/azure-developer-overview>. [Accedido: 22-may-2025].

**GRACIAS**

