

[Home](#)[About](#)[Content](#)[Others](#)

Manual Técnico – Clúster Dockerizado con Spark

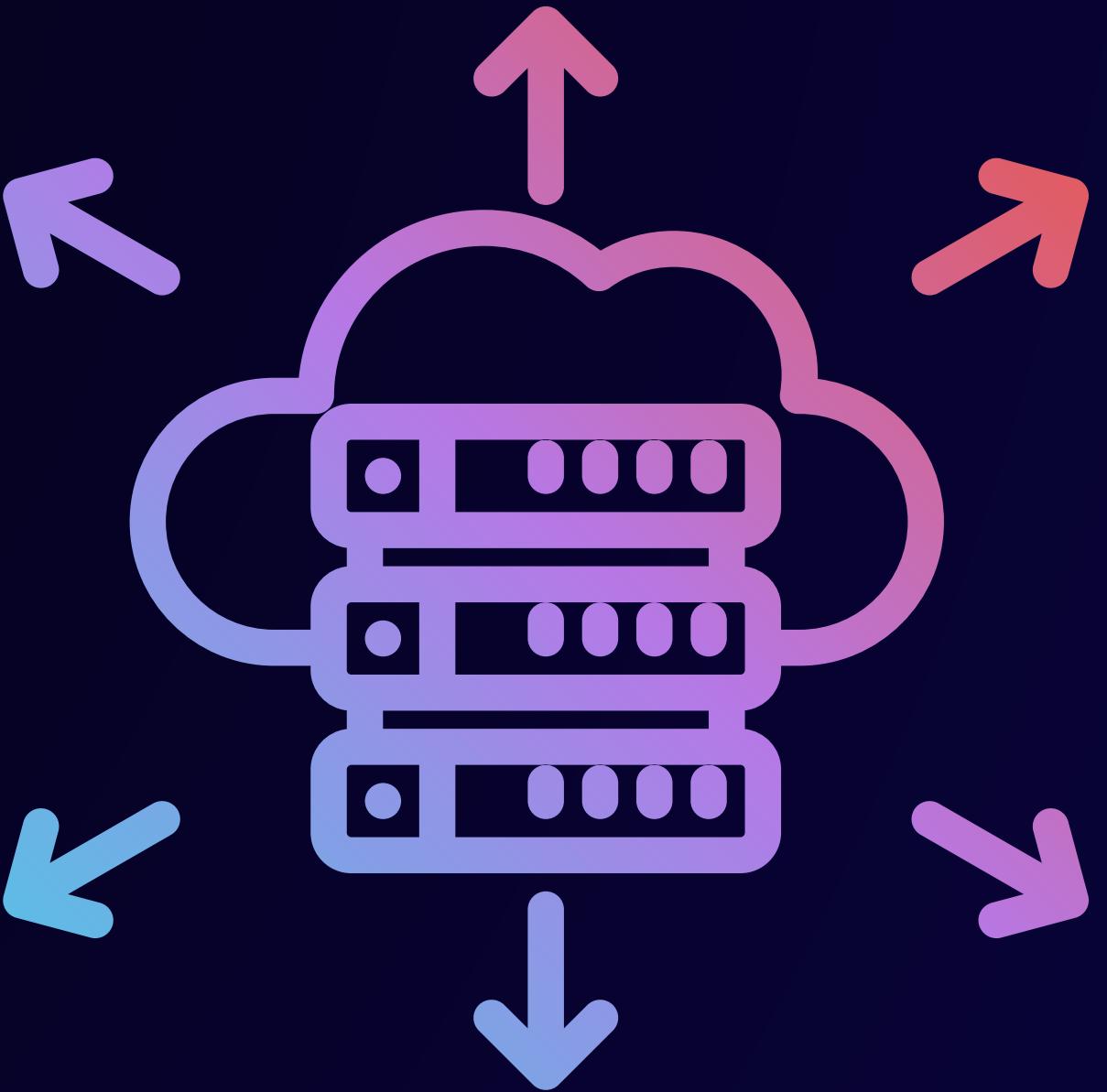
¿Qué es un clúster de computadoras?

Un clúster es un conjunto de computadoras (nodos) conectadas entre sí para trabajar como si fueran una sola máquina poderosa.

Permite ejecutar procesos en paralelo.

Optimiza el tiempo de ejecución de tareas pesadas.

Aumenta la disponibilidad de recursos para análisis de datos.

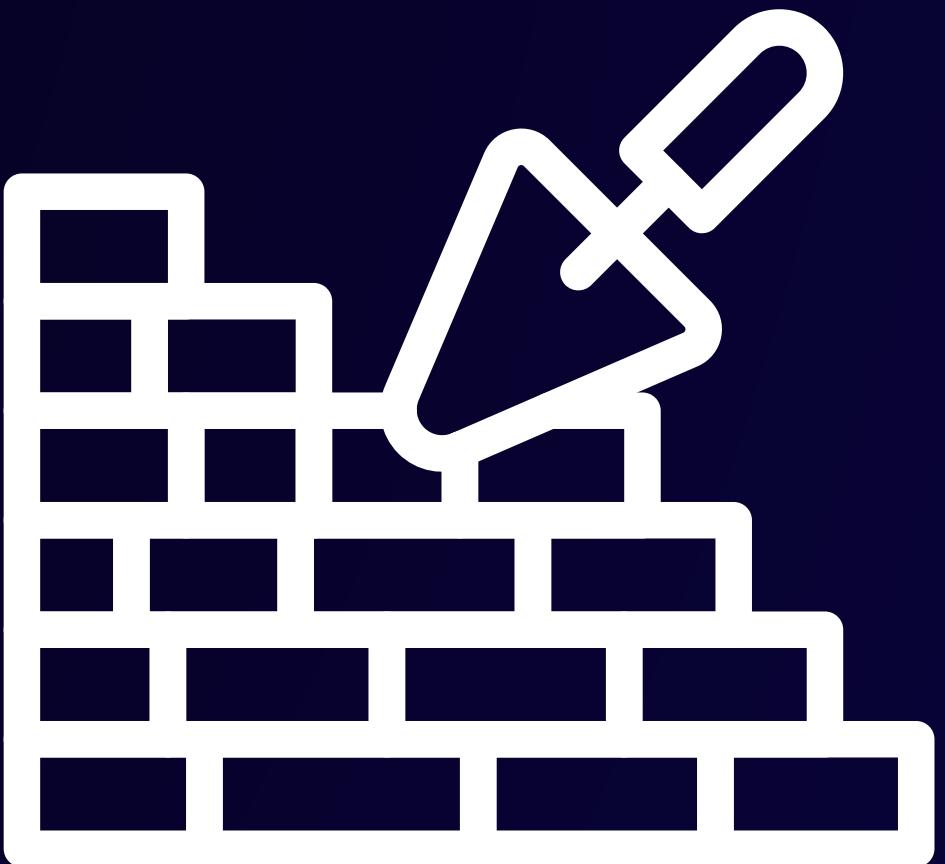


Arquitectura del Clúster

El clúster se construye con Docker swarm como herramienta de orquestación y Apache Spark para procesar

- **Nodo maestro:** Coordina y distribuye tareas
- **Nodos trabajadores:** Ejecutan contenedores de Spark
- **Red Docker:** Permite la comunicación entre contenedores
- **Balanceo de carga:** Docker swarm distribuye servicios de forma automática

Comunicación por red LAN (SWITCH + CABLES ETHERNET)



Tecnologías Utilizadas

- **Ubuntu LTS como sistema operativo base**
- **Docker para contenedores ligeros y portables**
- **Docker Swarm para orquestar el despliegue automático de servicios**
- **Apache Spark como plataforma de procesamiento distribuido**
- **NFS para almacenamiento compartido (opcional)**



REQUISITOS TÉCNICOS

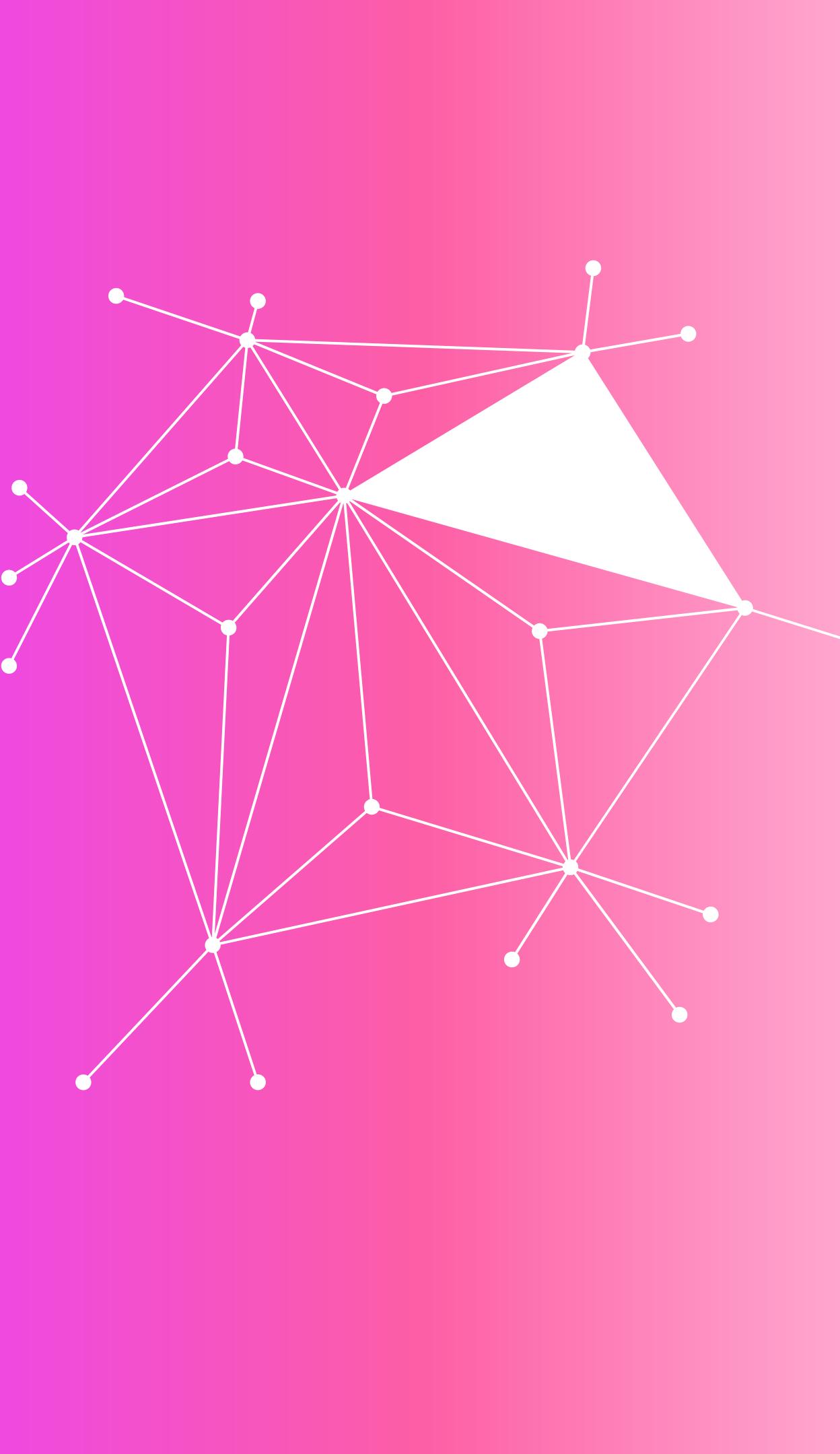
Hardware mínimo por nodo: 8-16 núcleos, 8-16 GB RAM

Almacenamiento: SSD, Carpetas compartidas

Red: IPs estáticas y switch de al menos 1 Gbps

Software: Instalación de Docker, Docker Compose, Apache Spark, NTP o timesyncd para sincronización horaria





Instalación y configuración

Proceso de instalación y configuración

Instalar Ubuntu y paquetes básicos

Instalar Docker y Docker compose en cada nodo

Configurar red (IP fija, Hosts, etc)

Iniciar Docker swarm desde el nodo maestro:
`docker swarm init --advertise-addr <IP_del_master>`

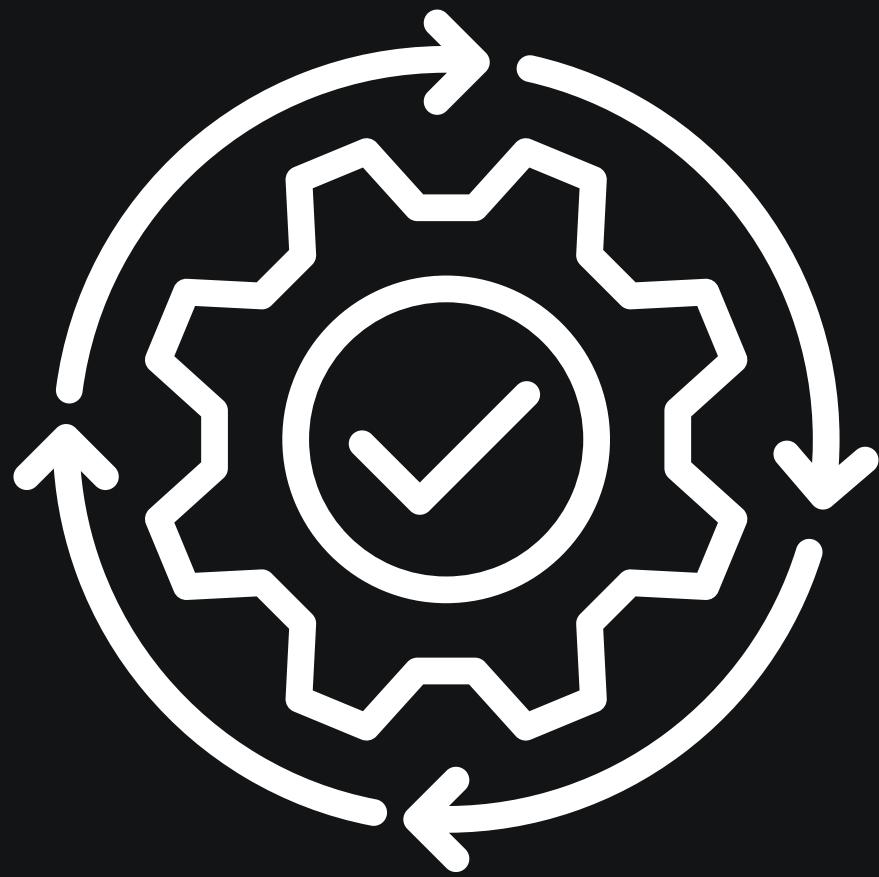
Unir los workers al swarm:
`docker swarm join --token <TOKEN> <IP_master>:2377`

Levantar servicios con docker-compose.yml
`docker stack deploy -c docker-compose.yml spark`

Pasos para levantar el Clúster

- Configuración de la red (asignación de IPs con DHCP)
- Levantar servicios (Docker compose)
- Inicializar Docker Swarm
- Verificar estado de los contenedores
- El master mueve el archivo CSV a la carpeta compartida
- Asignar roles y recursos a los nodos
- Acceder al panel Spark UI (<http://localhost:8080>)
- Correr código PySpark





Ejecución de Tareas con Spark

- DAG: Grafo dirigido acíclico.
- Transformaciones: (Lazy) map, filter, join.
- Acciones: (Ejecución) collect, show, count.
- Jobs: Unidades de trabajo.
- Stages: Etapas de ejecución.
- Tasks: Tareas individuales.
- Driver: Orquestador principal.
- Executors: Realizan las tareas.
- Cluster Manager: Asigna recursos (Docker).
- Optimización: Caching, shuffle, partición.

SEGURIDAD EN EL CLUSTER



- **Imágenes Seguras:** Confiables, mínimas, escaneo.
- **Redes Aisladas:** Puertos limitados, cifrado comunicación.
- **Spark:** Autenticación, autorización (ACLs), cifrado RPC/UI.
- **Secretos:** No hardcode, gestores de secretos.
- **Monitorización:** Logs, auditoría, alertas.
- **Parches:** Actualizaciones constantes.

Errores comunes y soluciones

ERROR

- **Spark no levanta**
- **Nodo no se une al swarm**
- **Falta sincronía horaria**
- **No se va la UI**

SOLUCIÓN

- **Revisar logs del contenedor y red Docker**
- **Verificar IP, token y conectividad**
- **Reinstalar y configurar NTP o usar timesyncd**
- **Asegurar que el puerto 8080 esté expuesto y no bloqueado por firewall**



Actividades en el Clúster

- Esta sección describe las operaciones clave en nuestro clúster Dockerizado con Spark:
- Despliegue de Aplicaciones: Enviar Jobs de Spark (`spark-submit`) al Master, que solicita contenedores Docker para los Workers. El script puede montarse con volumen o copiarse.
- Gestión de Recursos: Aumentar workers con `docker-compose up --scale spark-worker=X -d` o usar Docker Swarm para distribución.
- Monitoreo: Observar logs y el rendimiento en la Spark UI (`http://localhost:8080`).
- Interacción/Administración: Iniciar (`docker-compose up -d`), verificar (`docker ps`, `docker logs`), detener (`docker-compose down`) y limpiar (`docker system prune`) el clúster.
- Flujo de Datos: El `.py` puede montarse con volumen. Se realizan pruebas de procesamiento de jobs.



CONCLUSIONES

- El clúster dockerizado permite ejecutar tareas en paralelo de forma portátil y eficiente.
- Gracias a Docker y Spark, se logró simular un entorno distribuido real usando contenedores.
- Esta práctica consolidó conocimientos de redes, orquestación, procesamiento distribuido y automatización.
- El proyecto es escalable y adaptable a entornos académicos, experimentales o incluso productivos.