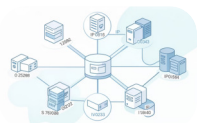
Proyecto Big Data 2025–2026 — Sistema de Control de Accesos NFC

Monitorización, procesamiento distribuido y visualización de datos. Equipo Azul — Módulo: Big Data Aplicado. Entorno: Cluster Ambari (CentOS 7). Fecha: febrero 2026.

Infraestructura y Conectividad (Resumen)

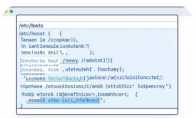


Topología

9 hosts en 172.16.200.x con nombres únicos (ambari3, ambari5, ambari8, ambari9, ambari10, ambari11, ambari12, ambari13, ambari15).

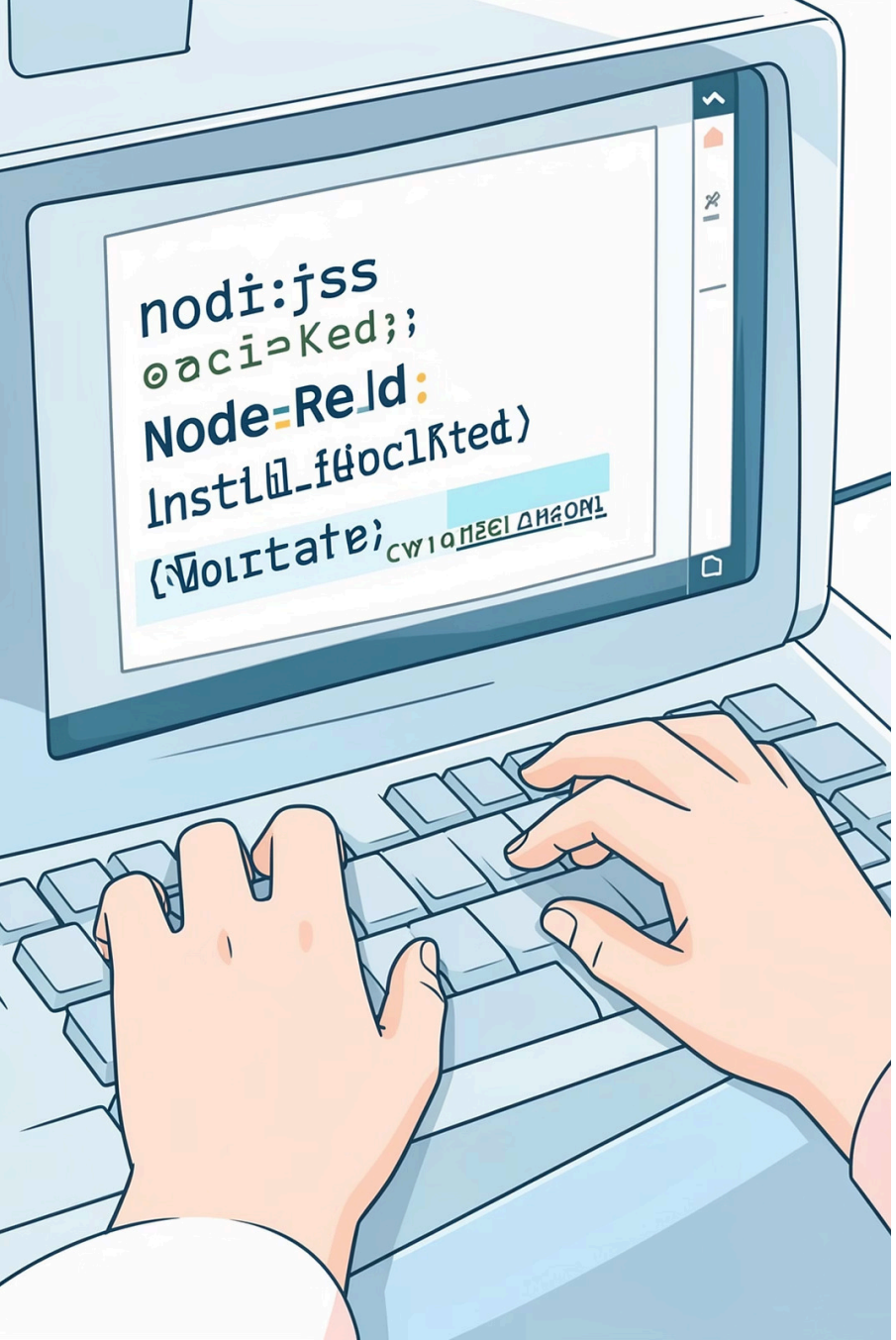
Adaptadores de red

Dos adaptadores por máquina: bridge para la red 172.16.200.x y NAT para acceso a Internet.



/etc/hosts

Contenido idéntico replicado en las 9 máquinas para resolución de nombres interna.



Node-RED en CentOS 7

VM ambari9: problema de compatibilidad con glibc 2.17; solución: instalar Node.js 16 (v16.20.2) y Node-RED 3.1.3. Pasos: descarga manual, configurar PATH, instalar npm global y verificar ejecución.

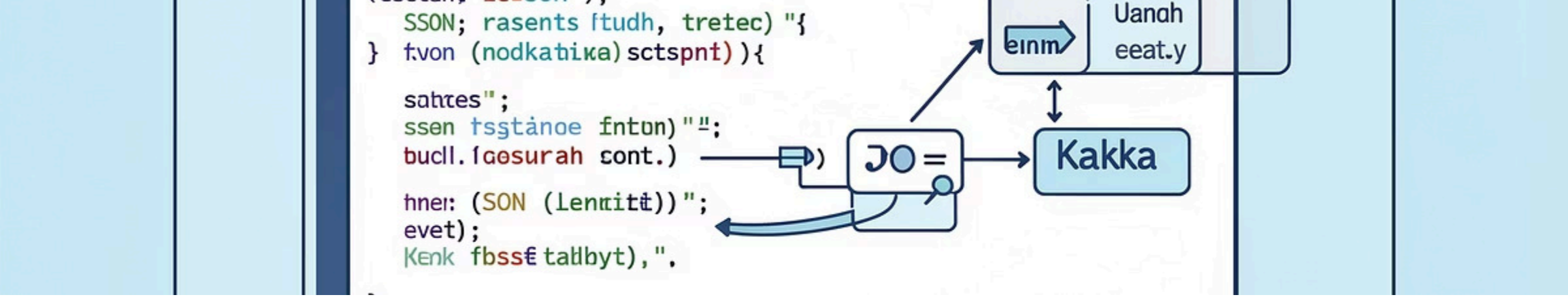
Arquitectura de Mensajería — Apache Kafka

Broker y topic

Broker configurado en nodo1 (/opt/kafka-2.8.1/config/server.properties) con listeners PLAINTEXT y advertised.listeners=172.16.200.28:9092. Topic: acceso-centros-nfc (3 particiones, replication-factor 1).

Parámetros clave

- broker.id=10
- zookeeper.connect=172.16.200.10:2181
- log.retention.hours=168



Productor NFC — Especificaciones

Lenguaje

Python 3.x

Broker

172.16.200.28:9092, 172.16.200.10:9092

Topic / Particiones

acceso-centros-nfc — 3 particiones

Frecuencia / Formato

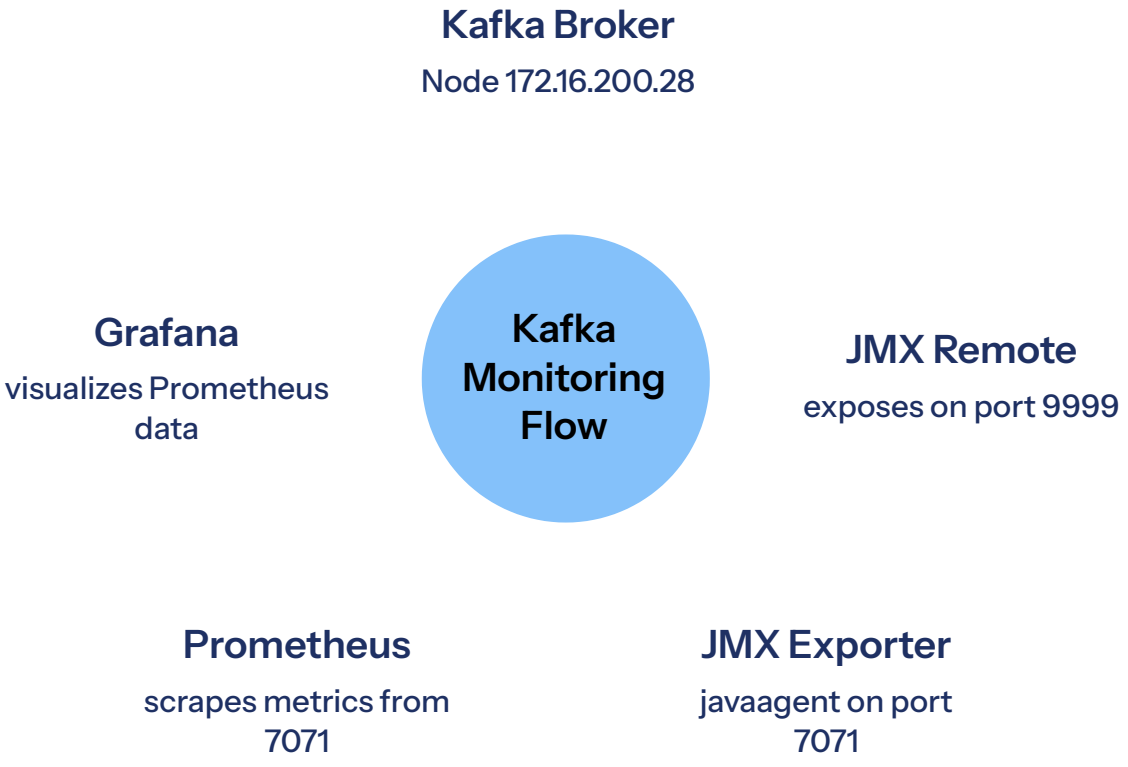
5 eventos/segundo — JSON UTF-8 — acks='all'

Estructura de un Evento NFC (Ejemplo)

Objeto JSON representativo con campos esenciales para control de accesos:


```
{
 "nfc_id": "NFC-A3F7D9E2C4B1",
 "timestamp": "2026-02-02T08:34:12.456789",
 "estudiante": {"nombre":"María García López","curso":"2º ESO"},
 "centro": {"nombre":"IES Miguel
Servet","codigo":"50008174","provincia":"Zaragoza"},
 "tipo_evento":"ENTRADA",
 "franja_horaria":"ENTRADA_MANANA",
 "punto_acceso":"Entrada Principal",
 "estado":"VALIDADO",
 "temperatura_corporal":36.5
}
```


Monitorización: JMX Prometheus Exporter



Configuración

nodo1 (172.16.200.28) expone métricas via javaagent jmx_prometheus (v0.20.0) en puerto 7071; JMX remoto en 9999. Regla de métricas adaptada en kafka-jmx-config.yml (nombres en minúscula).

Acción

Exportar métricas para Prometheus → paneles Grafana y alertas.

Almacenamiento y Bases de Datos

PostgreSQL

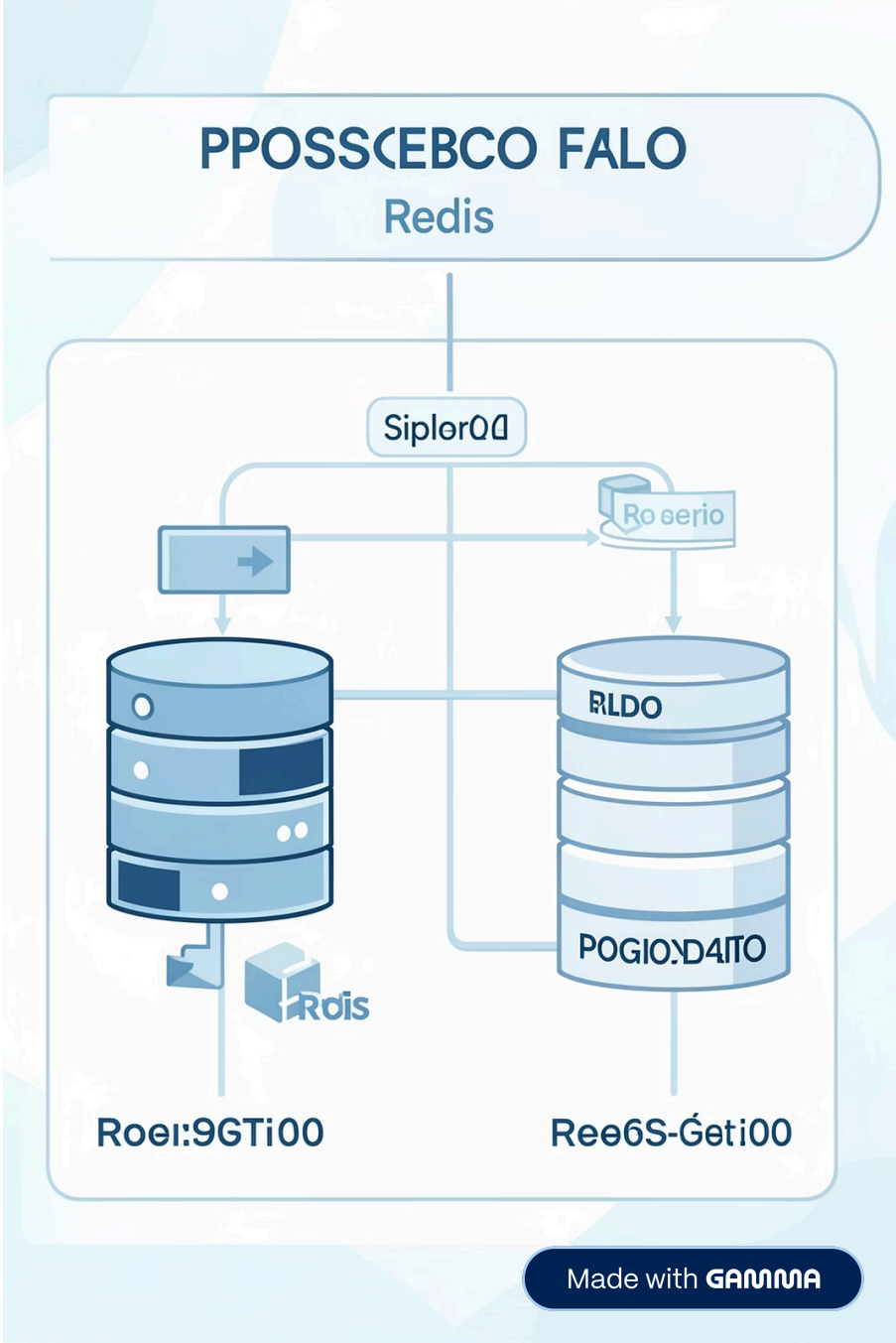
Crear usuario ambari_user, DB migasfree; editar pg_hba.conf para permitir acceso desde 192.168.1.103 y localhost.

Redis

Instancia con autenticación; comandos de verificación (AUTH, KEYS "*") y política de evicción (noeviction) configurada.

Caché y persistencia

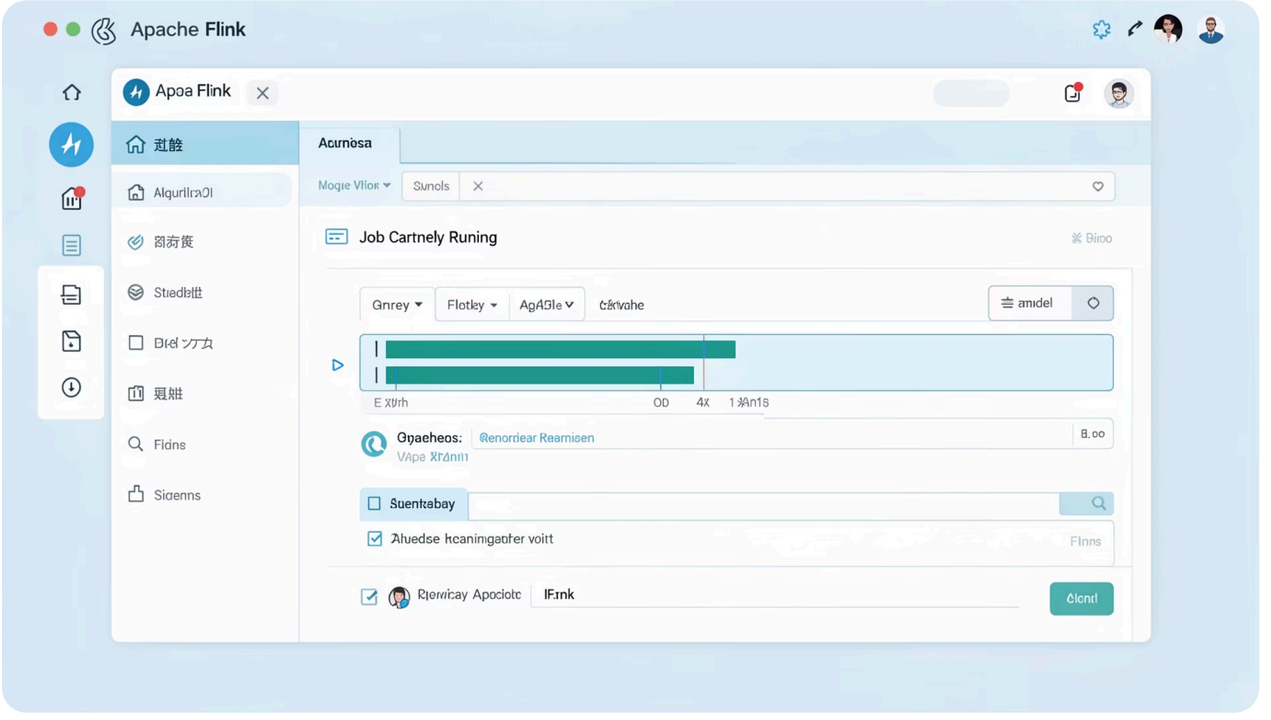
Redis como caché; PostgreSQL para persistencia transaccional.



Procesamiento y Notebooks

Apache Flink

Flink 1.15.4 (Scala 2.12). Maestro: ambari13. PyFlink instalado; pipeline con UDF que escribe en Redis (host 172.16.200.23).



Zeppelin

Conectividad JDBC con PostgreSQL (postgresql-42.5.0.jar), migración a Python 3 en intérpretes y sincronización de dependencias (pandas, matplotlib, seaborn).

Visualización, logros y próximos pasos

Apache Superset — Dashboard Migasfree

Componentes: pie de aplicaciones (414), barras por provincia (~100 máxima), treemap de centros (30), tabla de centros, comparativo entradas/salidas (~1.5k), ranking 10 centros más activos.

Logros

Pipeline en tiempo real, Kafka operativo, productor NFC robusto, integración Flink/Spark, monitorización (Prometheus/Grafana), visualización (Superset, Node-RED).

Próximos pasos

Implementar ML predictivo, escalar a más centros, optimizar rendimiento, alertas automáticas, añadir autenticación y APIs REST; integrar sensores físicos y generar reportes automáticos.