Tarea: Implantación de un Sistema Analítico Distribuido con Pinot

Contexto y objetivo general

En esta actividad vais a desplegar, configurar y poner en marcha un **sistema analítico distribuido** basado en **Apache Pinot**, ejecutándose sobre un clúster de máquinas virtuales en **AWS EC2**, con ingestión de datos en tiempo real mediante **Apache Kafka**.

El propósito de esta práctica es que podáis trabajar con una infraestructura que se parece mucho a la que utilizan empresas reales para análisis de datos con baja latencia: un ecosistema formado por **servicios coordinados**, tolerante a fallos, con ingestión *streaming* y consulta distribuida.

Al finalizar la tarea deberéis haber demostrado que sois capaces de:

- **Montar un clúster distribuido** de Pinot en AWS que separa plano de control, plano de datos y brokers de consulta.
- **Levantar un Kafka** de pruebas y producir datos.
- **Crear tablas en Pinot**, definidas con *schema* y *table config*.
- **Ingerir datos reales o generados** desde Kafka hacia Pinot.
- **Ejecutar consultas SQL** sobre el clúster y comprobar el reparto interno de los segmentos.
- **Analizar brevemente el comportamiento del clúster** (fallos de nodos, balanceo, latencias…).

Esta tarea representa un paso más con respecto a la que ya realizasteis en GitHub Code Spaces: ahora se busca trabajar con una infraestructura **multi-nodo real**, hosteada en servidores independientes, que puede sufrir fallos, reequilibrar datos y permitir consultas distribuidas.

Infraestructura a desplegar

Antes de comenzar la parte de ingestión y análisis, habrá que **montar una mini–infraestructura Big Data en AWS** con la siguiente topología:

1. ZooKeeper (3 nodos)

- Función: servicio de **coordinación distribuida**.
- Guarda metadatos del clúster.
- Determina qué nodos están vivos y quién actúa como líder.
- Maneja el consenso entre servicios.
- **Puertos:** 2181
- **Volúmenes:**
  - `/data` → datos internos del quorum
  - `/catalog` → metainformación para Pinot

El despliegue debe permitir que **si un nodo de ZooKeeper cae, el clúster siga operativo**.

2. Pinot Controller (2 nodos)

- Rol: el "cerebro" del clúster.
- Gestiona:
  - Catálogo de tablas
  - Segmentos y su ubicación
  - Rebalanceos
  - Schemas
  - Tareas de ingestión
- Depende de ZooKeeper para almacenar su estado.
- **Puertos:** 9001
- **Volúmenes:** `/controller`

La existencia de dos controladores permite practicar con **alta disponibilidad** del plano de control.

3. Pinot Broker (2 nodos)

- Rol:
  - Gestionan las consultas entrantes.
  - Localizan los segmentos necesarios.
  - Recogen respuestas parciales desde los servidores.
  - Devuelven el resultado fusionado al cliente.
- **Puertos:** 7001

Estos nodos permiten ejecutar *queries* distribuidas y comprobar cómo Pinot reparte la carga de trabajo.

4. Pinot Server (2 nodos)

- Rol:
  - Almacenan físicamente los **segmentos**.
  - Ejecución de las tareas pesadas de consulta.
  - Participan en reequilibrios.
- **Puertos:** 8001
- **Volúmenes:** `/server`

Es la parte "muscular" del sistema, donde realmente se ejecuta el cálculo.

Kafka y productor de datos

Además del clúster de Pinot, deberéis levantar un servicio **Kafka** (puede estar en uno o dos nodos) con un topic para pruebas. La ingesta se hará desde Kafka → Pinot mediante una tabla de tipo *REALTIME*.

Se pide:

- Crear un topic llamado **events**, **measurements**, **sensor-data** o similar.
- Producir datos de prueba. Podéis elegir:
  - Dataset generado con un script Python.
  - Dataset real (por ejemplo, sensores IoT simulados como ya hicimos en la tarea evaluable 1).
  - Mensajes aleatorios basados en JSON.

Debe quedar demostrado que Kafka **está recibiendo mensajes** y que Pinot es capaz de **consumirlos en tiempo real**.

Creación de esquemas y tablas en Pinot

Crear **al menos dos tablas** en Pinot, de cualquiera de estos tipos:

- **Una tabla REALTIME** conectada a Kafka.
- **Una tabla OFFLINE** generada desde un fichero CSV o JSON (opcional pero recomendado).
- O bien **dos tablas REALTIME**, si preferís trabajar exclusivamente con Kafka.

Consultas a ejecutar

Una vez ingestado el dataset, se deberán ejecutar varias consultas SQL a través de la interfaz de pinot, sea el controlador o el API del broker.

Consultas mínimas requeridas:

1. **Selecciones simples**

    - `SELECT * LIMIT 10;`
2. **Agregaciones**
    - `COUNT(*)`, `AVG(...)`, `MIN/MAX(...)`
3. **Filtrado por rangos de tiempo** (si aplica)
4. **Agrupaciones**
    - `GROUP BY`
5. **Consulta distribuida**
    - ver cuántos *segments* se han utilizado
    - comprobar si la consulta se ejecuta en ambos servers
6. **Pruebas de fallo**
    - detener temporalmente un broker o server y repetir la consulta
    - observar la resiliencia del sistema

El objetivo es que entiendan cómo Pinot ejecuta consultas en un entorno **realmente distribuido**.

Informe final a entregar

Entregar un informe breve (5–7 páginas aprox.) con:

1. Arquitectura desplegada

- Mapa de nodos con IPs privadas.
- Explicación del rol de cada servicio (ZooKeeper, controller, broker, server).
- Diagrama propio o generado.

2. Configuración

- Ficheros YAML/compose utilizados.
- Reglas de seguridad del AWS Security Group.
- Justificación de la elección de puertos y redes

3. Kafka

- Topic creado.
- Ejemplo de mensajes producidos.
- Captura demostrando recepción de datos.

4. Tablas en Pinot

- Los dos JSON completos (schema + table config).
- Explicación del modo de ingestión.
- Captura del estado de segmentos.

5. Consultas realizadas

- Al menos 6 consultas completas.

- Capturas reales mostrando los resultados.
- Interpretación de lo que muestra Pinot:
    - número de segments utilizados
    - número de servidores involucrados
    - latencias de respuesta

6. Prueba de resiliencia

- Detener un nodo y explicar:
    - qué se rompe, qué no
    - si el sistema sigue respondiendo
    - cambios visibles en la UI o en el cluster

7. Conclusiones

- ¿Qué dificultades habéis tenido?
- ¿Por qué empresas reales usan Pinot?
- ¿Qué ventajas tiene frente a Spark, Druid u otras soluciones?
- ¿Qué problemas veis en un despliegue como este?

Rúbrica general de evaluación

Apartado	%
Despliegue correcto del clúster en AWS	25%
Kafka funcional y productor enviando datos	15%
Creación correcta de schemas + tablas	20%
Ingestión correcta y estable	10%
Consultas SQL	15%
Pruebas de resiliencia y explicación	10%
Presentación y calidad del informe	5%

Ejemplos:

Os enlazo algunos ejemplos de clúster similar, lo podéis poner en marcha en GitHub antes de llevarlo a AWS:

https://github.com/startreedata/pinot-recipes/tree/main
https://github.com/startreedata/pinot-quickstart/tree/main

Otro en blog: Realtime data streaming kafka-flink-pinot-mysql