# Documentación del Pipeline de Ingesta: Datos de Steam (Datos.conf)

Este documento detalla la configuración del pipeline de Logstash diseñado para ingestar, procesar y enriquecer los datos de videojuegos de Steam desde un archivo NDJSON hacia un clúster de Elasticsearch seguro.

## 1. Fuente de Datos (Input)

El pipeline lee datos de un archivo local en formato **NDJSON** (Newline Delimited JSON).

* **Archivo fuente:** /home/g6/reto/datos/steam-games-data-vect.ndjson
* **Mecanismo de lectura:**
  + start\_position => "beginning": Garantiza que se lean todos los datos existentes si Logstash se reinicia sin registro previo.
  + sincedb\_path => "/dev/null": **Configuración de Desarrollo/Reto**. Fuerza a Logstash a "olvidar" su posición de lectura cada vez que se reinicia, re-ingestando todo el archivo desde cero. Esto lo hacemos porque los precios de los juegos pueden ir cambiando.
  + **Codec JSON:** Se utiliza codec => json directamente en el input. Esto pondria automáticamente cada línea del archivo en un objeto de evento de Logstash, eliminando la necesidad de un filtro json posterior.

## 2. Transformaciones y Lógica de Negocio (Filter)

Esta es la sección crítica donde se normalizan y enriquecen los datos.

### 2.1. Normalización de Fechas

* **Campo:** release\_date
* **Acción:** Se parsea el string de fecha (formato yyyy-MM-dd) a un objeto Timestamp real. Esto es crucial para poder filtrar por rangos de fechas en Kibana posteriormente.

### 2.2. Extracción de Datos No Estructurados (GROK)

Se utiliza **Grok** para estructurar el campo de texto libre pc\_requirements\_min. Este campo contiene especificaciones técnicas mezcladas en un solo string.

* **Objetivo:** Separar SO, Procesador, RAM, Gráficos y DirectX en campos individuales.
* **Patrones Utilizados:**
  1. **Patrón Completo:** Busca SO:, Procesador:, Memoria:, etc.
  2. **Patrón Fallback:** Busca lo mismo pero omitiendo SO:, para juegos que no especifican el sistema operativo en los requisitos mínimos.
* **Datos Específicas:**
  + %{DATA:min\_os}: Texto del SO.
  + %{DATA:min\_cpu}: Modelo del procesador.
  + %{NUMBER:min\_ram\_gb:int}:Número de la RAM y lo pasa a entero.
  + %{DATA:min\_gpu}: Modelo de la tarjeta gráfica.
* **Manejo de Errores:** Si el patrón no coincide, se añade la etiqueta grok\_pc\_fail, permitiendo identificar juegos con formatos de requisitos extraños.
* **Limpieza:** Se elimina el campo original pc\_requirements\_min para ahorrar espacio.

### 2.3. Conversión de Tipos y Limpieza (Mutate)

Elasticsearch necesita tipos de datos estrictos para realizar agregaciones (sumas, promedios) correctamente.

* **Conversiones:**
  + price\_eur, price\_initial\_eur, vector\_embedding -> **Float** (para cálculos decimales).
  + discount\_pct, metacritic\_score, etc. -> **Integer**.
  + is\_free -> **Boolean**.
* **Renombrado (Normalización):**
  + steam\_id pasa a ser appid (estándar de Steam).
  + price\_eur pasa a ser price\_final (claridad semántica).
* **Eliminación de Ruido:** Se borran campos metadatos de Logstash (@version, host, path) y el campo message (que ya fue procesado), reduciendo el tamaño del documento final.

### 2.4. Enriquecimiento de Datos (Ruby)

Se utiliza código Ruby para aplicar lógica de negocio que mutate no puede manejar.

* Lógica de Categorización de Precios:  
  Se crea el campo price\_category basado en el precio final:
  + **Gratis:** Si is\_free es true o el precio es 0.
  + **Barato:** Precio < 15.0 EUR.
  + **Normal:** Precio < 40.0 EUR.
  + **Premium:** Precio >= 40.0 EUR.
  + *Uso:* Esto facilita crear gráficos de "tarta" (Pie Charts) en Kibana agrupados por categoría económica.

### 2.5. Auditoría de Datos

* **Campo:** last\_updated
* **Acción:** Se duplica el @timestamp de ingestión en un campo explícito last\_updated para tener constancia de cuándo se procesó el dato.

## 3. Salida y Almacenamiento (Output)

Los datos procesados se envían al clúster de Elasticsearch.

### 3.1. Indexación Dinámica

* **Configuración:** index => "steam-games-%{+yyyy.MM.dd}"
* **Explicación:** Se genera un índice nuevo cada día (ej. steam-games-2025.12.10). Esto es una buena práctica para la gestión del ciclo de vida de los datos (ILM), permitiendo borrar o archivar índices antiguos fácilmente.

### 3.2. Seguridad y Conexión

* **Hosts:** Se conecta a 3 nodos del clúster (192.199.1.53, .65, .66) para alta disponibilidad.
* **Autenticación:** Usa **API Key** en lugar de usuario/contraseña.
* **Encriptación:**
  + ssl\_verification\_mode => "full": Verifica que el certificado del servidor sea válido y confiable.
  + ssl\_certificate\_authorities: Ruta a la CA (http\_ca.crt) para validar los certificados de los nodos.

### 3.3. Idempotencia (Prevención de Duplicados)

* **Configuración:** document\_id => "%{appid}"
* **Importancia Crítica:** Al forzar que el ID del documento en Elasticsearch sea el appid del juego (que es único), podemos re-ejecutar el pipeline tantas veces como queramos.
  + Si el juego ya existe, se **actualiza** (no se duplica).
  + Si no existe, se crea.