

# Spark

Luis Ramirez Monterosa

<https://github.com/lramirez/jug-spark>

# Big Data

Big Data permite procesar data

Variable

Veloz

Volumen



# Big Data

Big Data permite procesar data

Variable

BD relacionales, BD nosql

Veloz

Volumen



# Big Data

Big Data permite procesar data

Variable

BD relacionales, BD nosql

Veloz

cargas ETL son ahora en tiempo real

Volumen



# Big Data

Big Data permite procesar data

Variable

BD relacionales, BD nosql

Veloz

cargas ETL son ahora en tiempo real

Volumen

Terabytes al día, incluso Petabytes



# Aplicación típica de Big Data usando Kafka



# Kafka no es una base de datos



Tópico de salida produce 1M de trabajos diarios

Se ingresan a base de datos 10%

---

# Kafka es un log con expiración



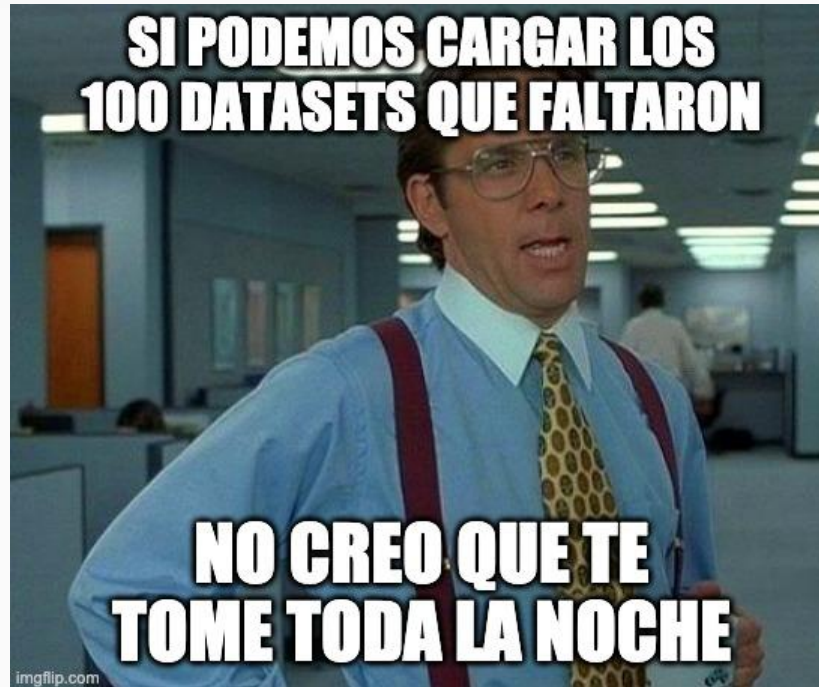
Valor expiración por default: 7 días

Cargar la data es costoso en bases altamente relacionales

---



Tu jefe a las 4:00 pm



# La vida del desarrollador a las 4:00 pm

- El tópico de resultados fue respaldado
- Un archivo de texto de 30GB
- Cada línea es un JSON
- 2M de records asociados a un dataset
- La llave “datasetId” esta en el JSON
- Hay 2000 datasets
- Solo es necesario cargar 100 datasets

# La vida del desarrollador a las 4:00 pm



- El tópico de resultados fue respaldado
- Un archivo de texto de 30GB
- Cada línea es un JSON
- 2M de records asociados a un dataset
- La llave "datasetId" esta en el JSON
- Hay 2000 datasets
- Solo es necesario cargar 100 datasets

A close-up photograph of a person's hands writing on a whiteboard. The person is wearing a dark long-sleeved shirt. The background is blurred, showing some bokeh lights. The text "La solución" is overlaid in white on the left side of the image.

# La solución

Spark requiere  
10 líneas de código

# Qué es Spark?



**Spark es una herramienta de Big Data general capaz de procesar datasets en archivos (locales y nube), bases de datos, unirlos, filtrarlos, manipularlos.**

**Puede ser procesado de forma local, un solo nodo**

**Procesar trabajos en un cluster  
con resiliencia  
escalable horizontalmente**

# Cuantos commits de mis empleados existen en Github

Los eventos en github tienen actores cuyo id es único por persona.

Filtrar por actor.id

Hay 8,440 empleados de mi empresa

Hay 94,950 committers

```
root
|-- actor: struct (nullable = true)
|   |-- avatar_url: string (nullable = true)
|   |-- gravatar_id: string (nullable = true)
|   |-- id: long (nullable = true)
|   |-- login: string (nullable = true)
|   |-- url: string (nullable = true)
|-- created_at: timestamp (nullable = true)
|-- id: string (nullable = true)
|-- org: struct (nullable = true)
|   |-- avatar_url: string (nullable = true)
|   |-- gravatar_id: string (nullable = true)
|   |-- id: long (nullable = true)
|   |-- login: string (nullable = true)
|   |-- url: string (nullable = true)
```

# Commits en

- wget <http://data.githubarchive.org/2015-03-01-{0..23}.json.gz>
  - 24 archivos
  - Formato JSON
  - Cada linea es un record
  - Hay 438,018 records
-



# Demo

.....

```
7 ▶ def main(args : Array[String]): Unit = {
8
9     // Spark session available as 'spark'.
10
11     val spark = SparkSession.builder().appName( name = "githubarchive")
12         .master( master = "local[*]").getOrCreate()
13
14     // Spark context available as 'sc'
15     val sc = spark.sparkContext
16
17     def myEmp: Set[String] = Source.fromFile("/Users/luisramirez/prj/github-archive/employees-id.csv").getLines.toSet
18     val broadcastEmployees = sc.broadcast(myEmp)
19     val isInEmployee = anId => broadcastEmployees.value.contains(anId)
20     val isEmp = spark.udf.register( name = "SetContainsId", isInEmployee)
21
22     val allRecs = spark.read.json( path = "/Users/luisramirez/prj/github-archive/*.json")
23
24     import spark.implicits._
25
26     val filtered = allRecs.filter(isEmp($"actor.id"))
27
28     filtered.repartition( numPartitions = 1).write.json( path = "./out/myemps")
29
30 }
```

# Conclusión



