Programa de Arquitectura para entornos de Big Data & Cloud

BIG DATA ACADEMY

Alonso Melgarejo

Big Data aplicado a la empresa

HAGA CLIC PARA AGREGAR TEXTO

¿Big Data aplicado a las empresas?

BIG DATA EMPRESARIAL

BIG DATA ACADÉMICO

Conceptos

- ¿Computación cuántica?
- ¿Teleportación de datos?

$\cdot\ sq(C_t, M_t) - cp\ .\ hc(C_{t-1}, C_t)$

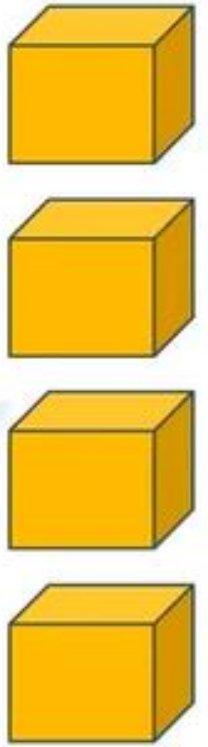
Big Data

Pliego conceptual "estable" de Big Data

- Batch
- Real Time
- Machine Learning
- Infraestructure
- Tuning
- ...

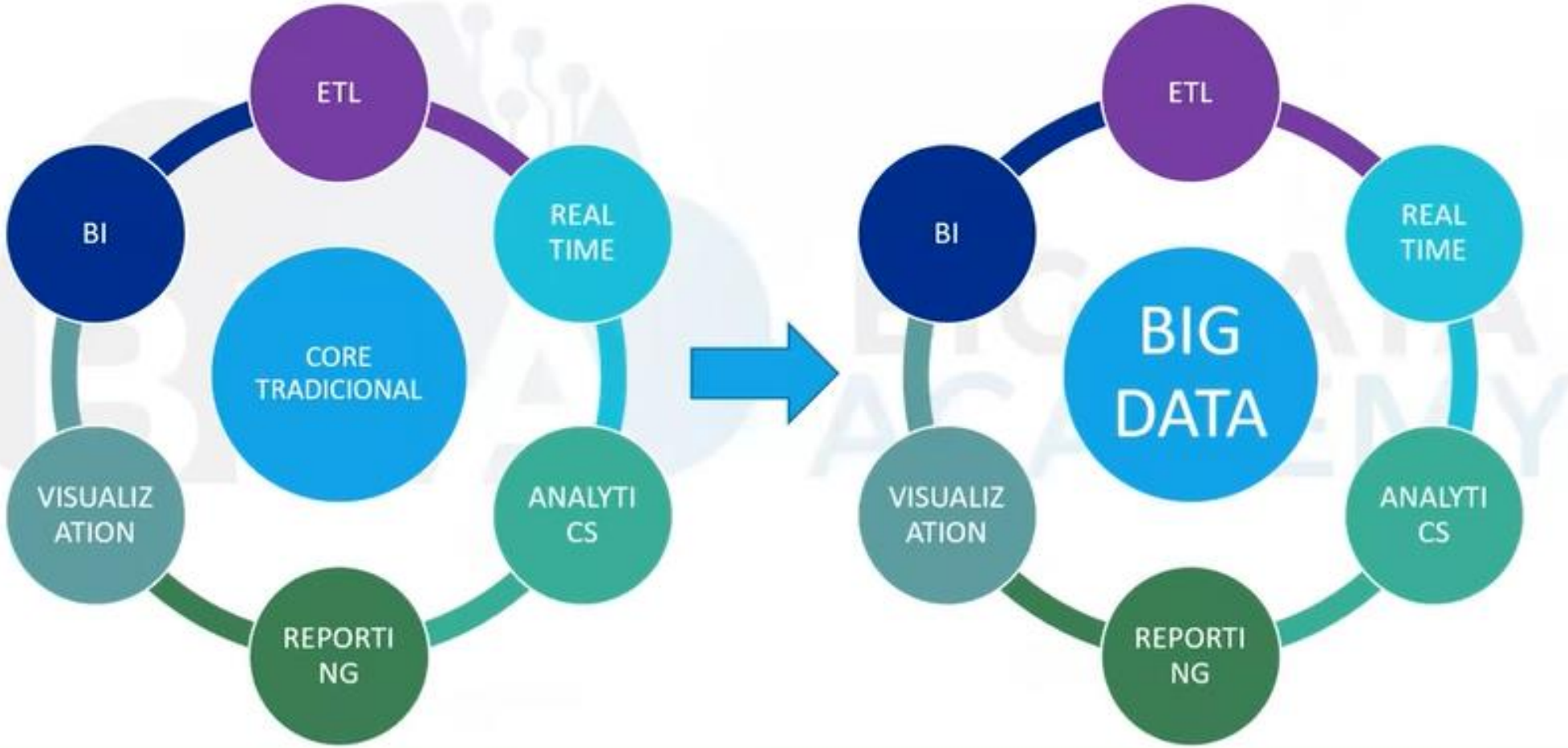
- Tecnologías
- Estándares
- Patrones
- Arquetipos

Buenas prácticas en la industria



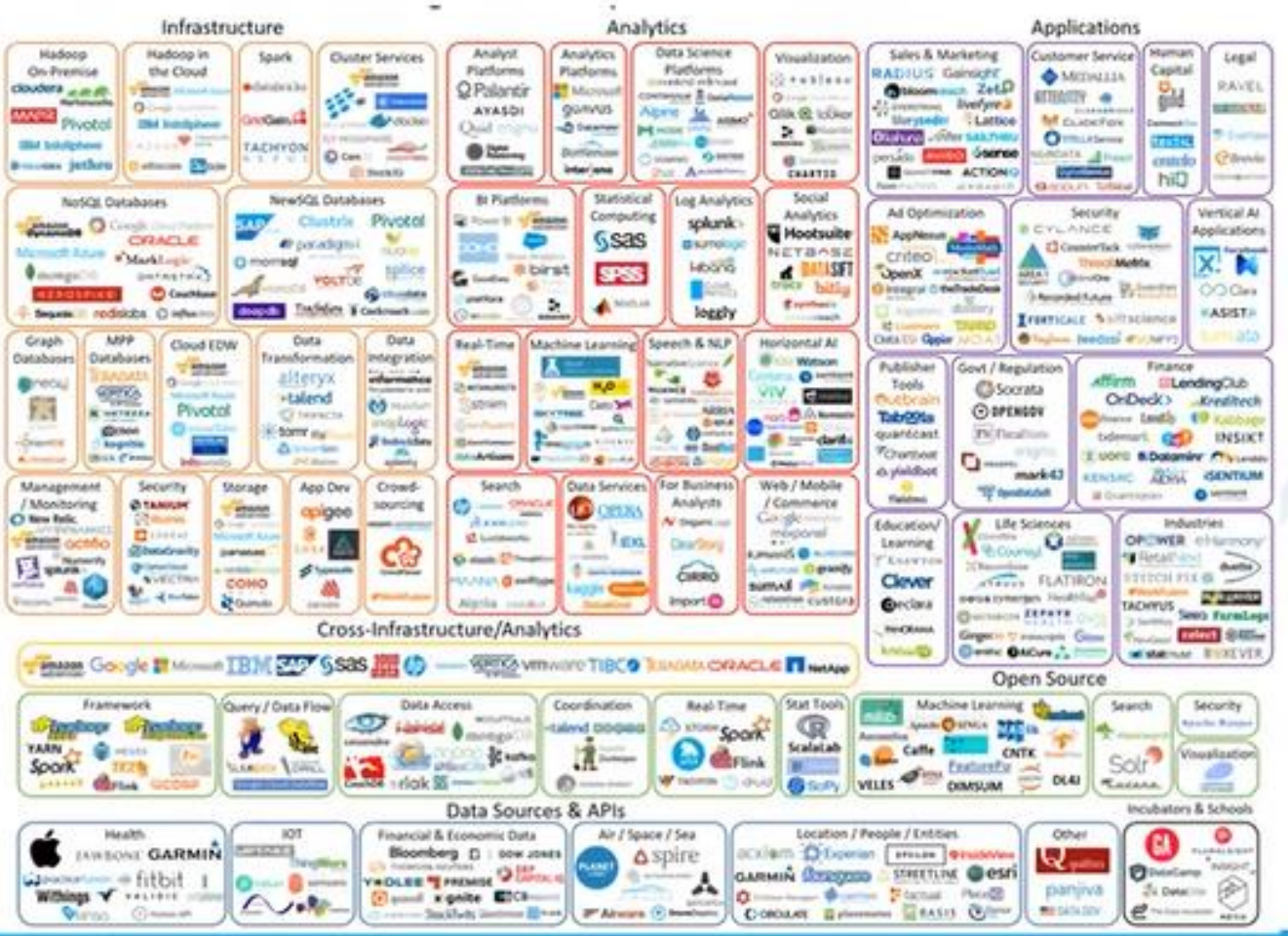
Soluciones Empresariales

Concepción de un Proyecto Big Data



Tecnologías en el mundo del Big Data

En 2016



Tecnologías en el mundo del Big Data

En 2020

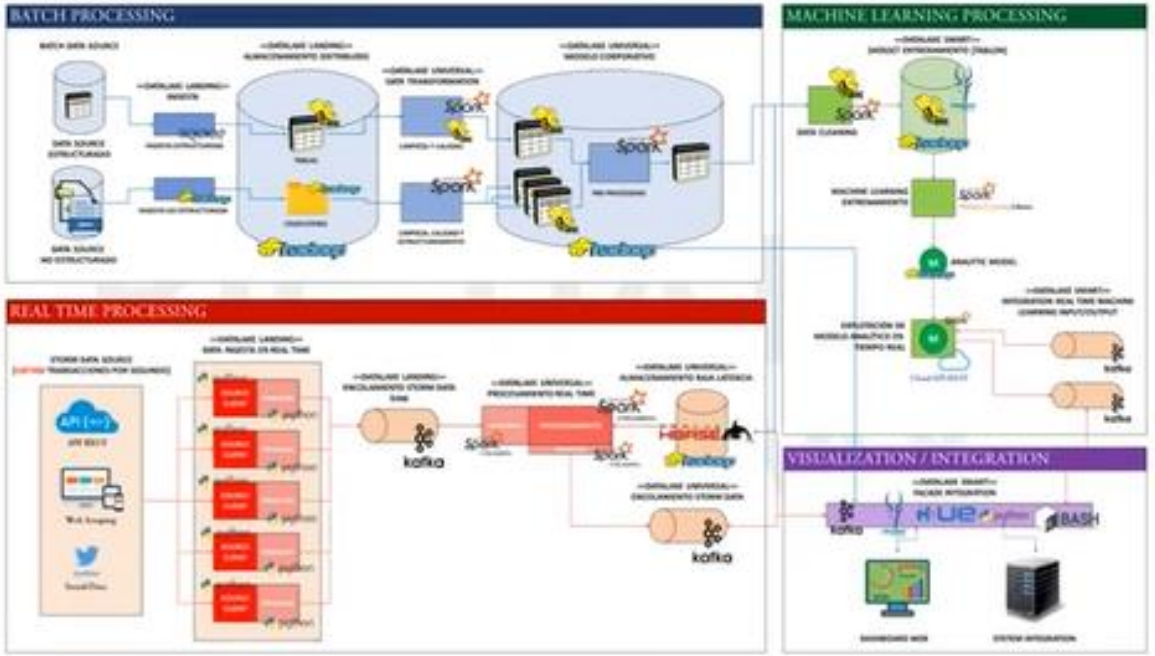
Propuesta de mix tecnológico

Definición arquitectónica

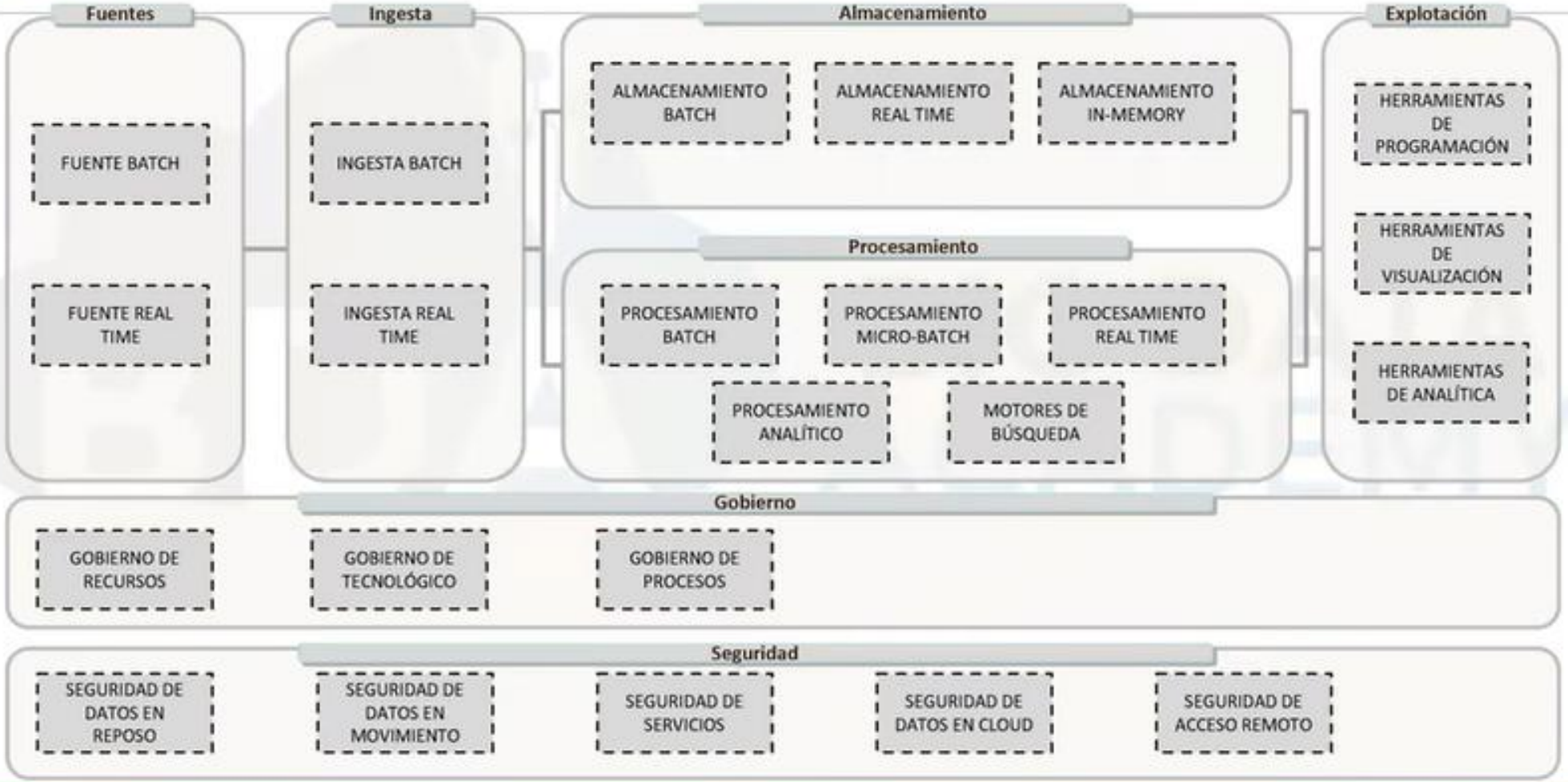
HAGA CLIC PARA AGREGAR TEXTO

Vistas arquitectónicas

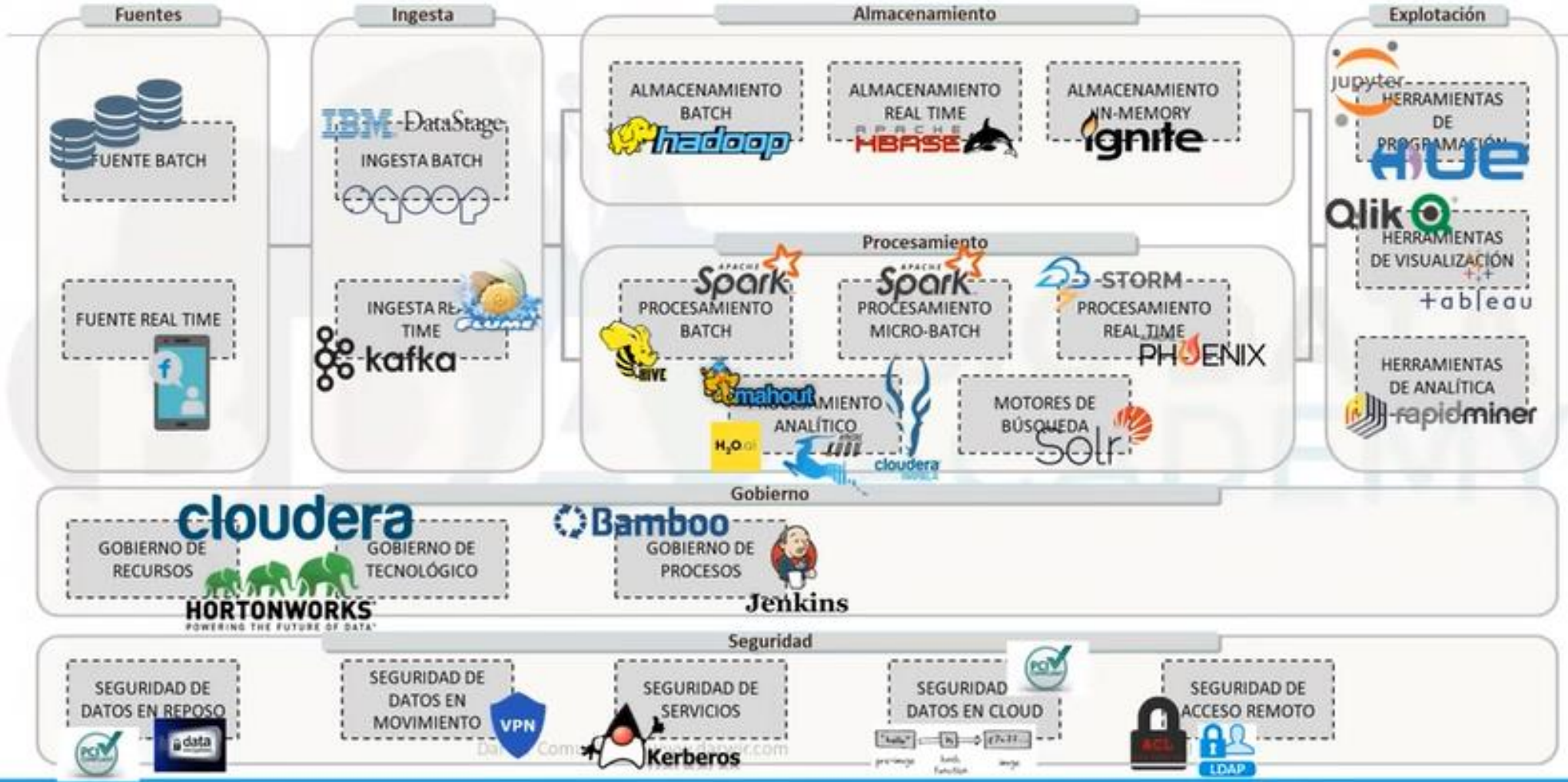
- Vista conceptual
- Vista tecnológica
- Vista de patrones de diseño
- Vista de infraestructura
- Vista de gobierno
- Vista de seguridad
- Vista de modelamiento, calidad, devops, ...



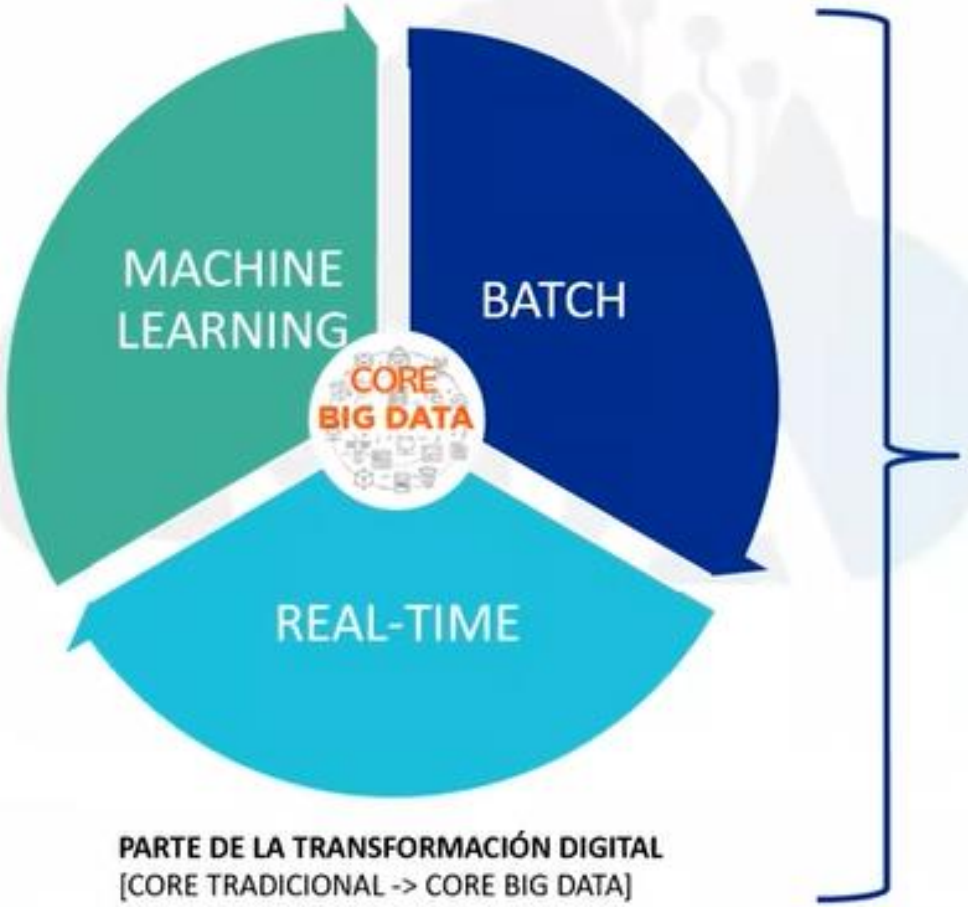
Arquitectura conceptual general de Big Data



Arquitectura tecnológica



Vistas de una arquitectura de Big Data



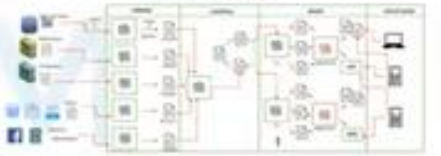
PARTE DE LA TRANSFORMACIÓN DIGITAL
[CORE TRADICIONAL -> CORE BIG DATA]

VISTAS DEL CORE DE BIG DATA

- Vista de arquitectura
- Vista de gobierno
- Vista de infraestructura

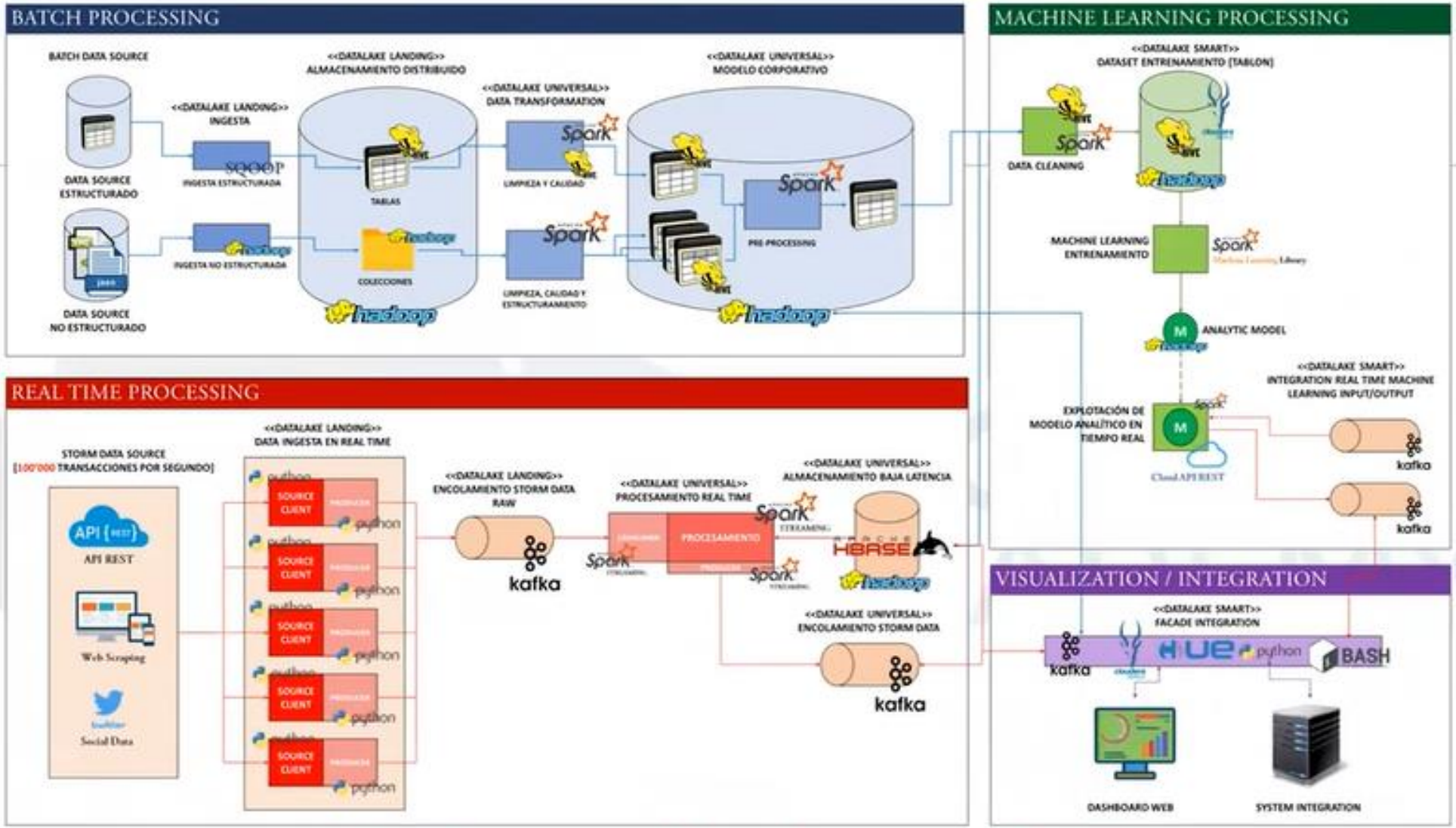
Conceptual y tecnológica

Datalake



On-Premise y Cloud

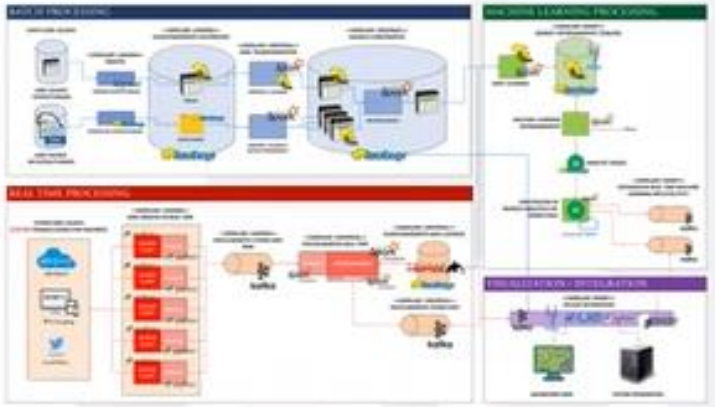
Vista Conceptual con Patrones de Diseño de Big Data



¡Este diagrama sí lo podemos codificar!

Vista de Infraestructura

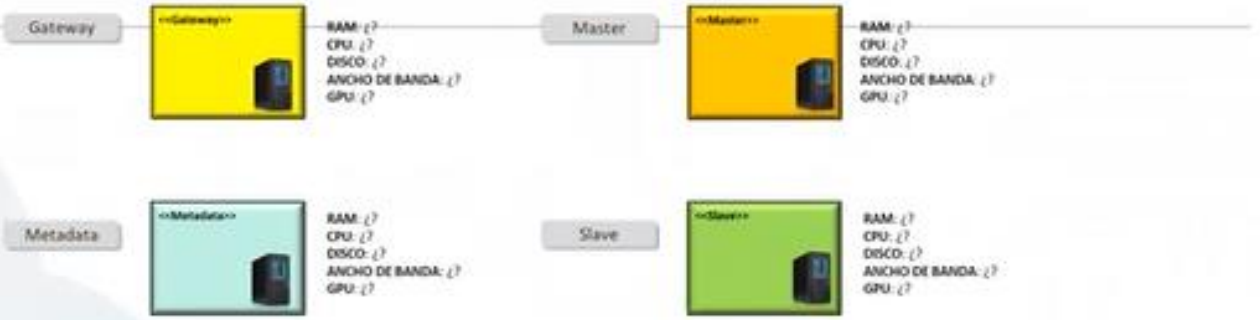
Arquitectura tecnológica



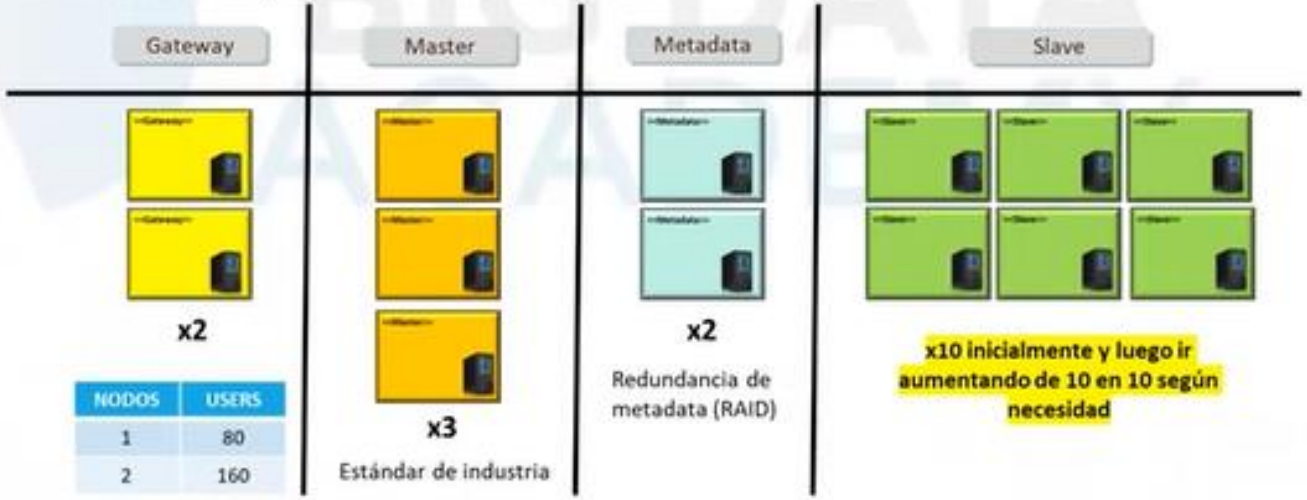
Debemos de traducir la necesidad tecnológica a Hardware:

- On-Premise
- Cloud IaaS
- Cloud PaaS
- Cloud SaaS

¿Qué características Hardware necesitamos?: RAM, CPU, DISCO, Ancho de Banda, GPU

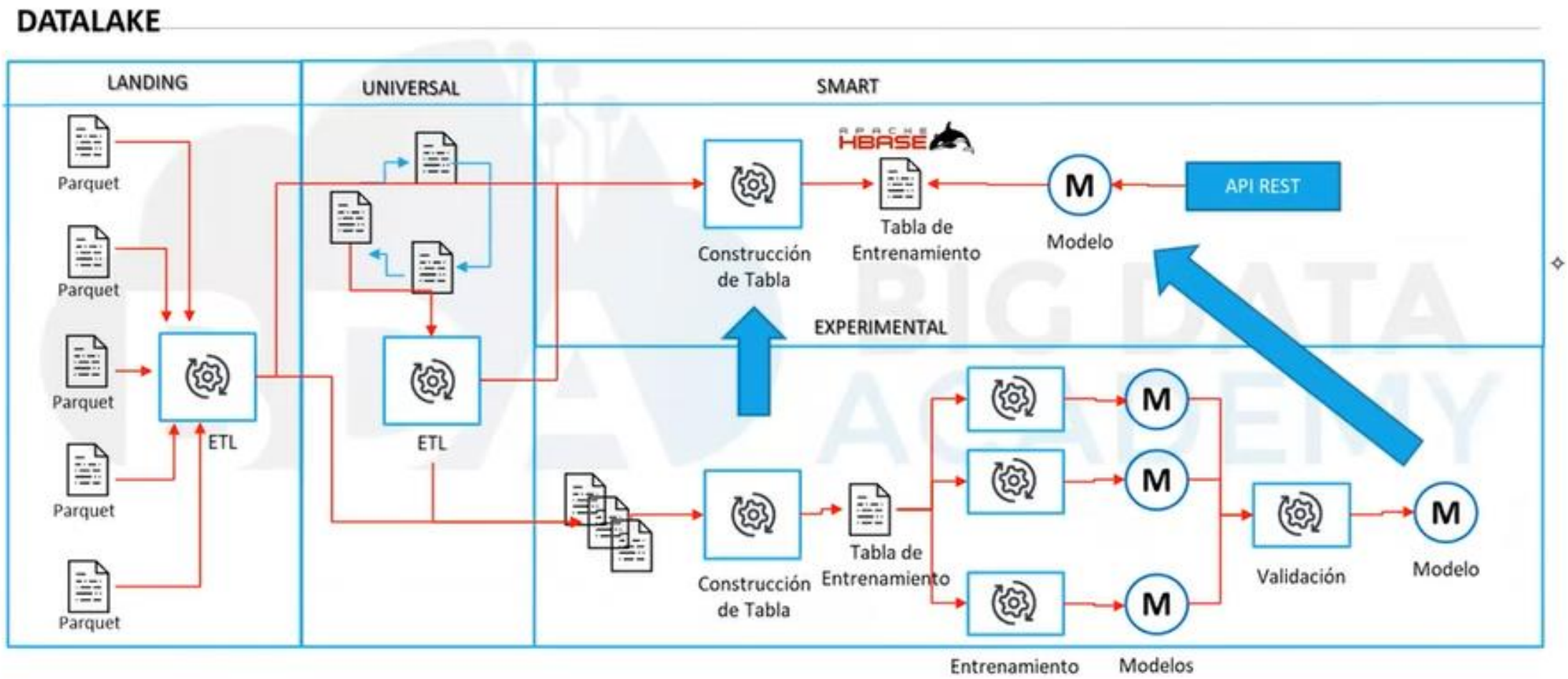


¿Cuántos roles y servidores necesitamos?

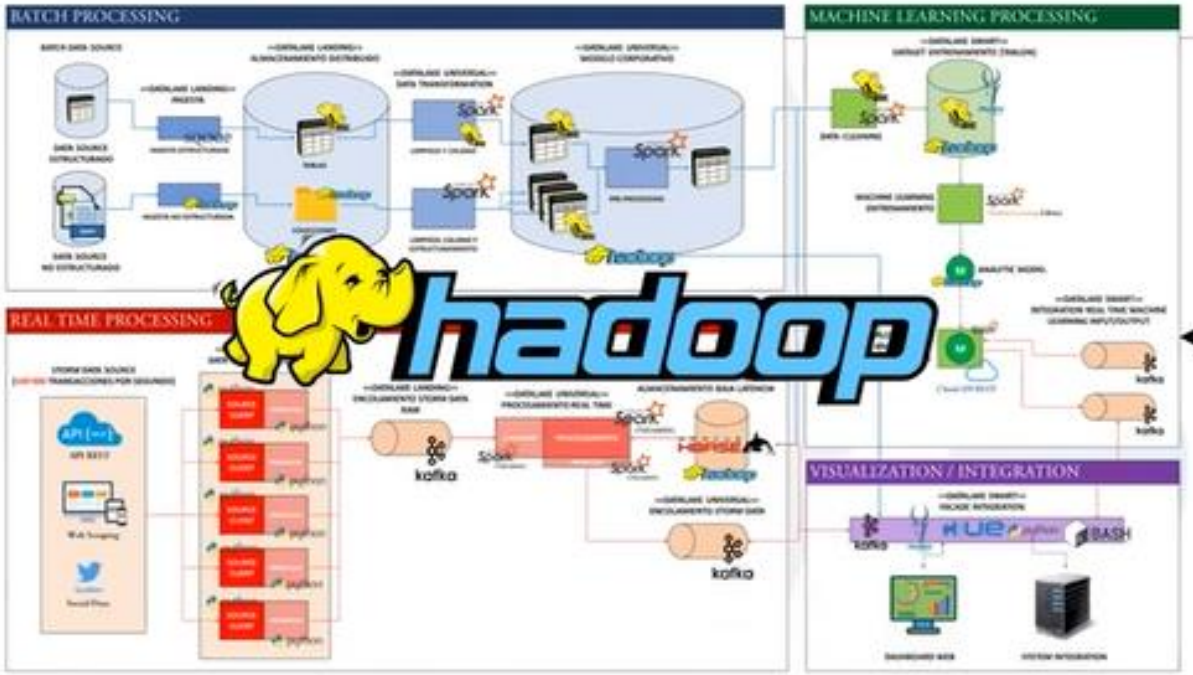


NODOS	USERS
1	80
2	160

Vista de Gobierno



Vista Tecnológica



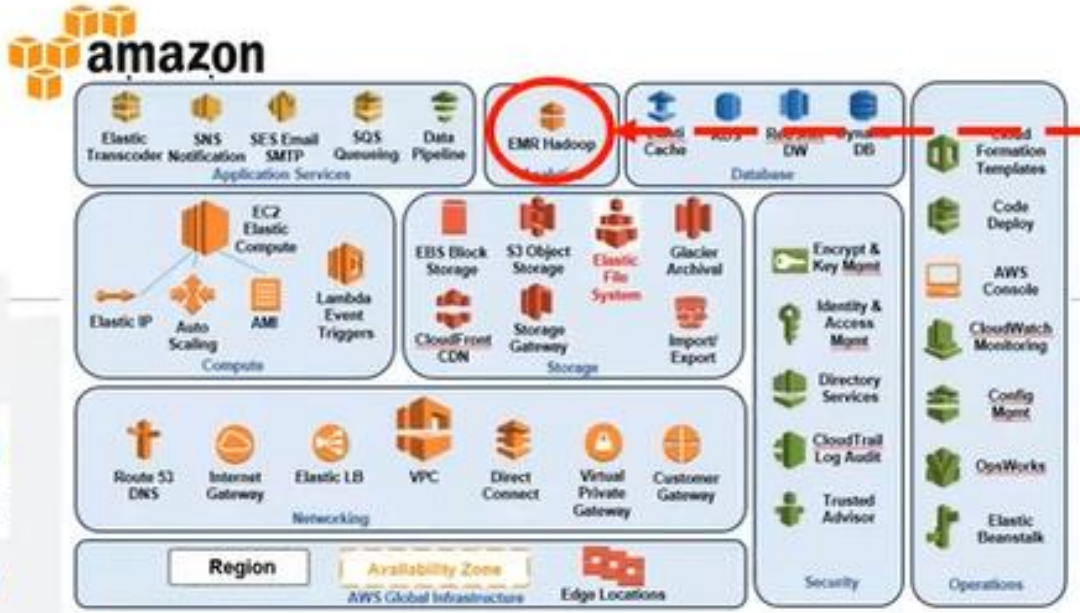
Basado en el Ecosistema **Estándar Hadoop**

Los patrones de diseño son agnósticos al ecosistema tecnológico

¿Arquitecturas tecnológicas Cloud?

HAGA CLIC PARA AGREGAR TEXTO

Servicios Cloud

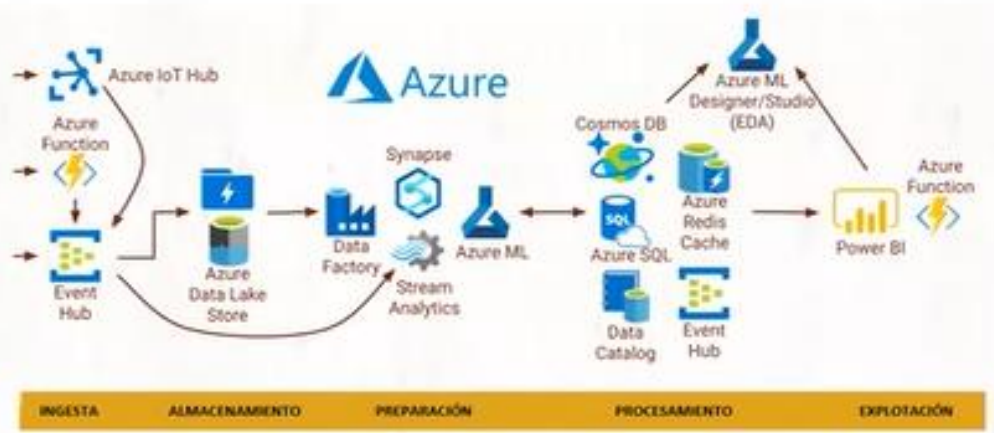


¡Existen muchos servicios de Cloud!

Lo que nos importan son los Servicios de Big Data on Cloud

El flujo del Big Data on Cloud

Cada nube ofrece su propio mix tecnológico,
pero **el flujo de procesamiento es estándar y agnóstico a la nube:**



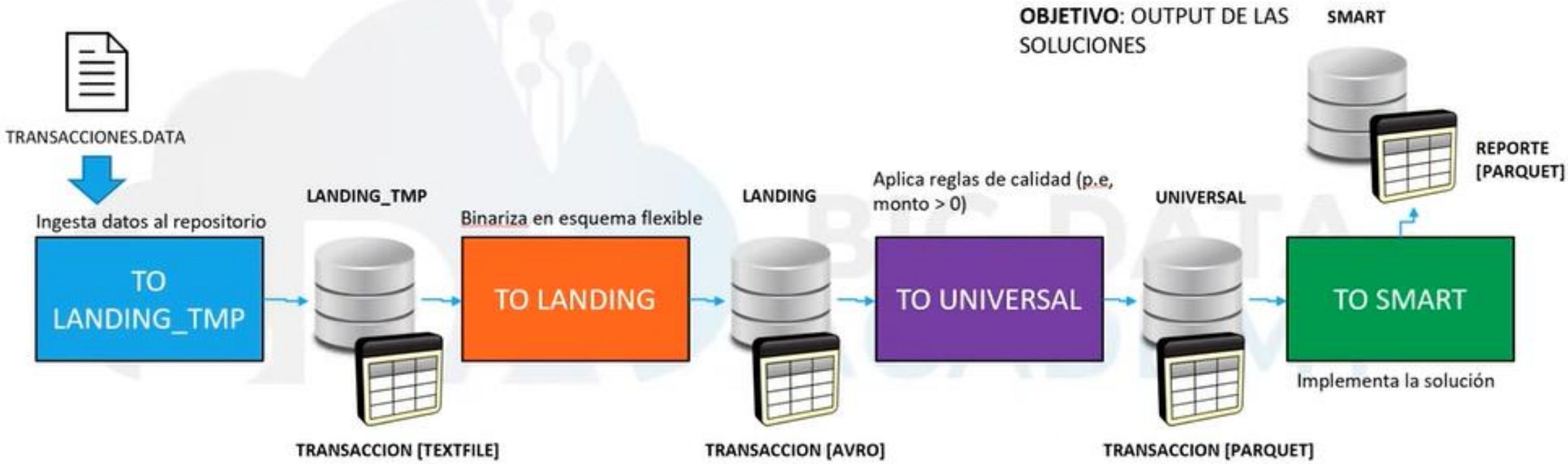
Herramientas estándar

Almacenamiento	
Interfaz SQL	
ETL	
Soluciones	
Workflow	

Herramientas cloud equivalentes

	ECOSISTEMA ESTÁNDAR	aws	Google Cloud Platform	Azure
Almacenamiento	hadoop	amazon S3	Cloud Storage	Azure Blob Storage
Interfaz SQL	HIVE	Amazon Athena	Google Big Query	Azure Synapse Analytics
ETL	HIVE	AWS Glue	Data Fusion	DATAFLOW
Soluciones	APACHE Spark	amazon EMR Spark	Cloud Dataproc Spark	Spark
Workflow	OOZIE	Airflow aws	Google Cloud Composer	Azure Data Factory

Flujo estándar sobre un Datalake



OBJETIVO: CAPTURAR DATOS

OBJETIVO: ESQUEMAS FLEXIBLES
¿Qué pasa si mañana "transacciones" incluye un campo más?

OBJETIVO: DATOS LIMPIOS CON ESQUEMA FIJO DE RÁPIDO PROCESAMIENTO

Traducción de estándares a cada cloud de cualquier flujo que llegue hasta UNIVERSAL

Estándares sobre un clúster agnóstico (P.E. cloudera):

Data capture
Binarization
Data catalog
Datalake
Checkpoint
...

¿Y qué pasa con las soluciones?

Transaccion en “Universal”

```python
#Importamos el objeto para instanciar una sesion
from pyspark.sql import SparkSession

#Instanciamos una sesion
#Dependiendo de la potencia del cluster, tomara hasta 3 minutos en asignarle recursos
spark = SparkSession.builder.getOrCreate()

#Leemos los datos que procesaremos
dfTransaccion = spark.read.format("parquet").load("/objects/inputs/transaccion")

#Almacenamos el dataframe como vista temporal
dfTransaccion.createOrReplaceTempView("dfTransaccion")

#Procesamos el dataframe
dfReporte = spark.sql("""
    SELECT
        T.ID_PERSONA ID_PERSONA,
        COUNT(T.ID_PERSONA) CANTIDAD_TRANSACCIONES,
        SUM(T.MONTO) MONTO_TRANSACCIONES
    FROM
        dfTransaccion T
    GROUP BY
        T.ID_PERSONA
""")

#Guardamos el resultado en un directorio temporal del cluster
dfReporte.write.mode("overwrite").format("parquet").save("/objects/outputs/reporte")
```

Reporte resultante en “Smart”

Codificación agnóstica en SPARK sobre las tres nubes

Pueden ser soluciones más complejas como Redes Neuronales

Transaccion en “Universal”

```
###
 # @section Aplicamos modelos de Deep Learning [RED NEURONAL]
 ##

#Importamos el algoritmo
from pyspark.ml.classification import MultilayerPerceptronClassifier

#Configuramos el algoritmo
algoritmo3 = MultilayerPerceptronClassifier(
    maxIter=1000,
    layers=[9, 50, 50, 2],
    blockSize=128,
    seed=1234,
    labelCol='Survived',
    featuresCol='features'
)

#Generamos el modelo con el algoritmo seleccionado
modelo3 = algoritmo3.fit(dfTrain)

#Validamos el modelo
dfPredicciones3 = modelo3.transform(dfTest)
dfPredicciones3.show()
```

Red Neuronal

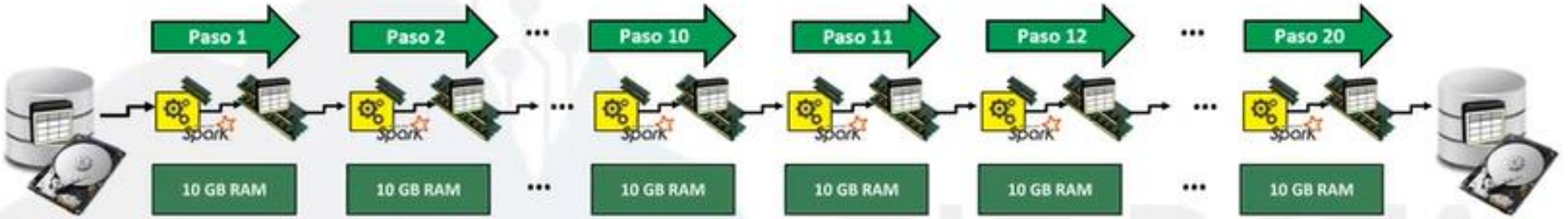
Codificación agnóstica en SPARK sobre las tres nubes

¡La implementación de nuestras soluciones **sí puede ser agnóstica**!

Vista de patrones de diseño [Proceso BATCH]

HAGA CLIC PARA AGREGAR TEXTO

Problema: Mi script tiene muchos pasos y no tengo suficiente RAM en el clúster



[PASO 1 -> PASO 10]
100 GB RAM

```
... 48 elided
Caused by: java.lang.OutOfMemoryError: Java heap space

scala> 20/06/09 08:28:58 ERROR TaskContextImpl: Error in TaskCompl
java.lang.IllegalStateException: Block broadcast_24 not found
        at org.apache.spark.storage.BlockInfoManager$$anonfun$2.ap
```

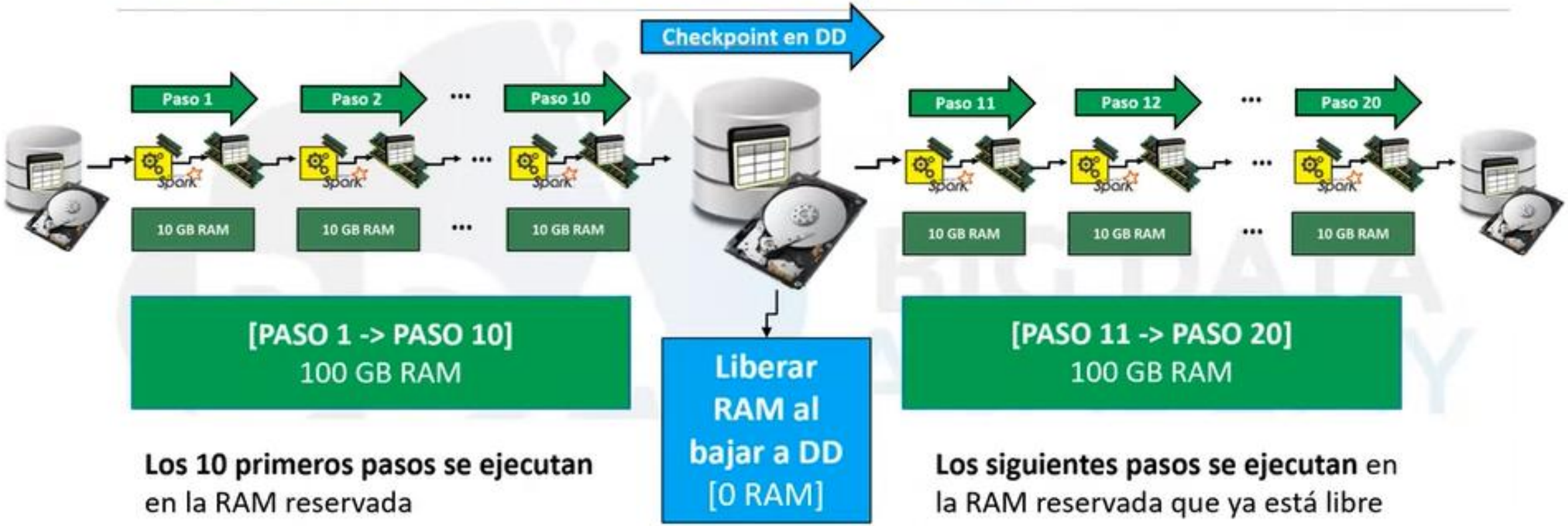
ERROR

Los 10 primeros pasos se ejecutan en la RAM reservada

No hay más RAM para los demás pasos y el proceso colapsa

Si nuestro proceso requiere de muchos pasos y ya no tenemos RAM, **¿cómo podemos programarlo sin que colapse?**

Solución: Patrón checkpoint



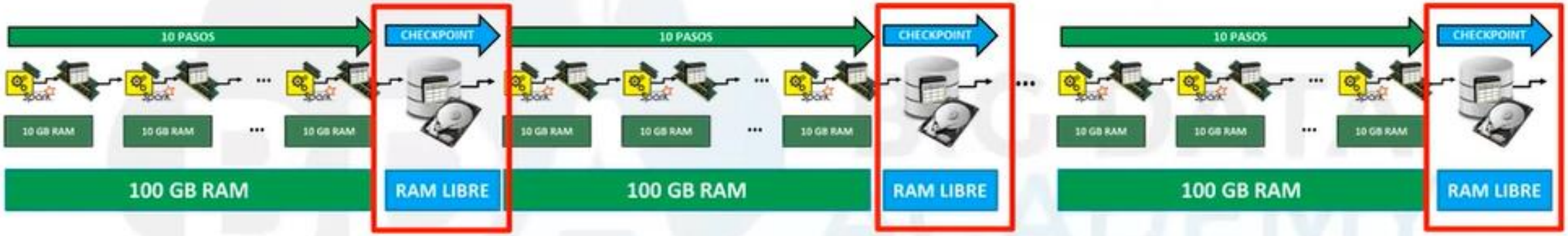
Los 10 primeros pasos se ejecutan en la RAM reservada

Los siguientes pasos se ejecutan en la RAM reservada que ya está libre

Almacenamos en disco duro el dataframe del paso 10 para liberar toda la ram de los dataframes del paso 1 al paso 9 que lo generaron, leemos nuevamente el dataframe y continuamos procesándo

¿Y si aumentamos más pasos en el proceso?

Cuando la memoria RAM esté a punto de llenarse, crearemos un checkpoint, de esa manera nuestro proceso podrá tener todos los pasos necesarios



Gracias al "checkpoint", **evitamos que la memoria colapse.**

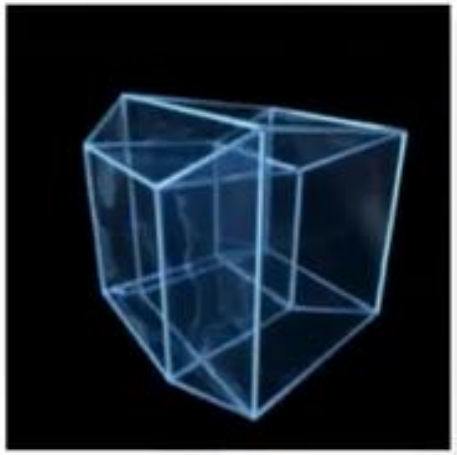
¿Y tiene alguna **desventaja**?: Sí, ya que **agrega un paso de escritura al disco duro de la resultante de la cadena de procesos**

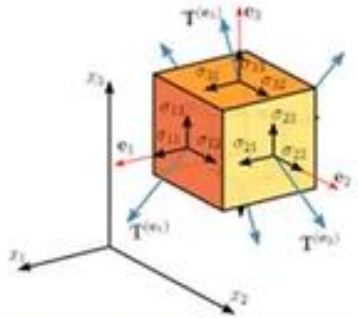
Vista de patrones de diseño [Proceso MACHINE LEARNING]

HAGA CLIC PARA AGREGAR TEXTO

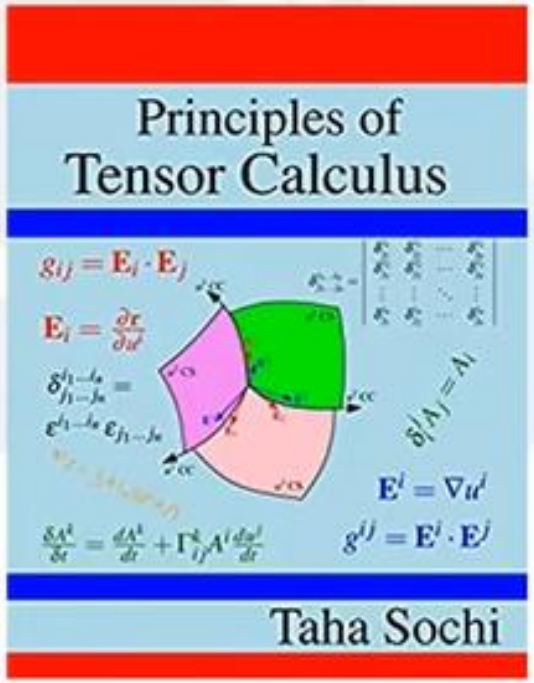
¿Qué es un tensor?

Objetos hiper-dimensionales

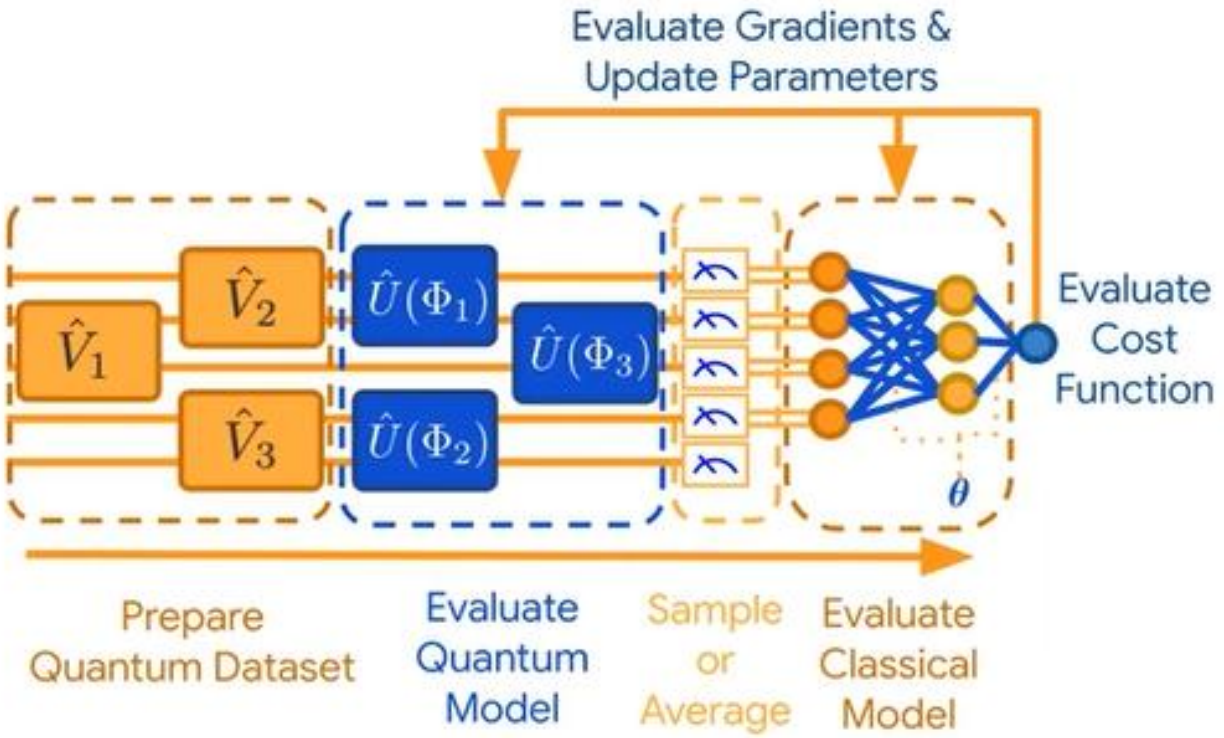




Cálculo tensorial



Aplicaciones para computación cuántica

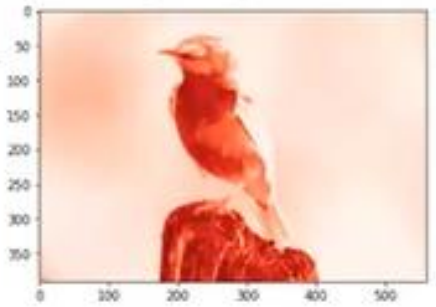


Pongamos un ejemplo simple para entender el concepto

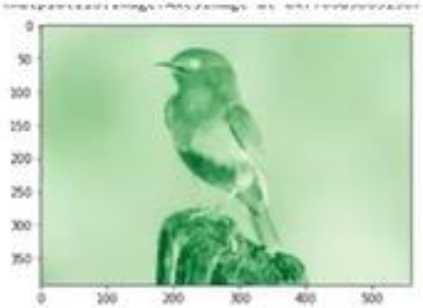
¿Qué es un tensor?

Imagen original

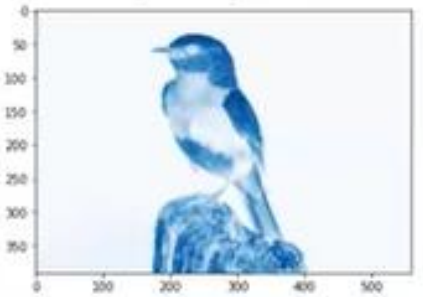
Representación tensorial
[alto, ancho, canal]



Canal R



Canal G



Canal B

Representación tensorial de una imagen

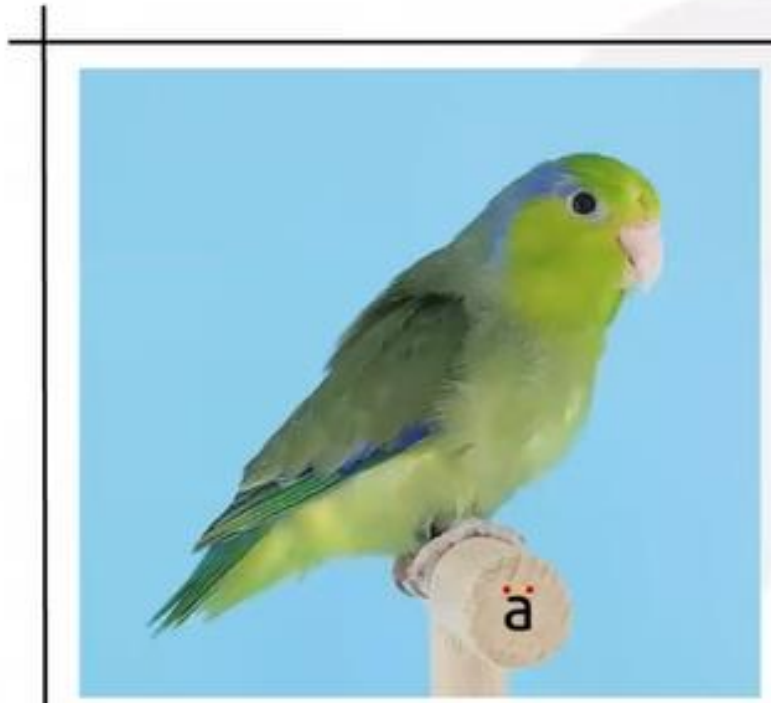


IMAGEN RAW
[Tipo: JPG]
[Canales: RGB]
[Tamaño: 32x32 píxeles]

Feature Engineering

CANTIDAD DE FEATURES
32x32x3 = 3072 [x1, x2, … x3072]

Cada color se representa por un número entre 0 y 255

CANAL R	CANAL G	CANAL B
17 19 23 35 … 45	32 22 76 86 … 33	22 65 11 38 … 21
81 23 11 23 … 87	18 94 70 67 … 54	94 42 75 30 … 98
…	…	…
54 24 45 32 … 56	45 67 55 39 … 48	55 32 65 71 … 92

REPRESENTACIÓN TENSORIAL
[1 Matriz para cada canal]
[Cada matriz de 32x32]

POR EJEMPLO
Pixel[PosX = 2, PosY = 1] = [R = 11, G = 70, B = 75]

Datasets para procesamiento de imágenes en Deep Learning

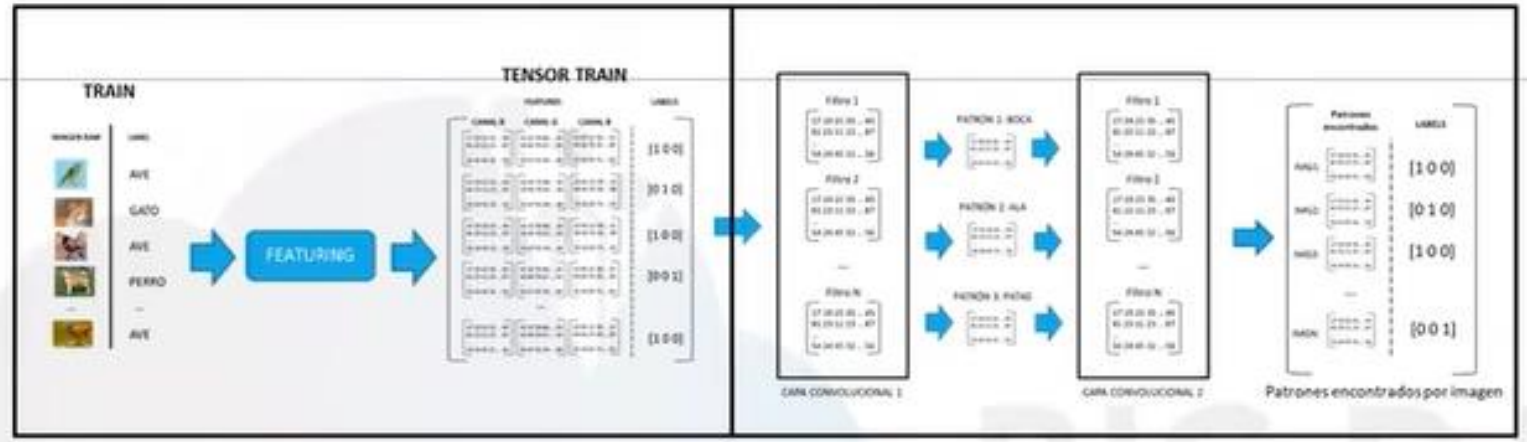
IMAGEN RAW	LABEL
	AVE
	GATO
	AVE
	PERRO
...	...
	AVE

Dataset con 500 mil imágenes

FEATURES			LABELS
CANAL R	CANAL G	CANAL B	
17 19 23 35 ... 45 81 23 11 23 ... 87 ... 54 24 45 32 ... 56	32 22 76 86 ... 33 18 94 70 67 ... 54 ... 45 67 55 39 ... 48	22 65 11 38 ... 21 94 42 75 30 ... 98 ... 55 32 65 71 ... 92	[1 0 0]
17 19 23 35 ... 45 81 23 11 23 ... 87 ... 54 24 45 32 ... 56	32 22 76 86 ... 33 18 94 70 67 ... 54 ... 45 67 55 39 ... 48	22 65 11 38 ... 21 94 42 75 30 ... 98 ... 55 32 65 71 ... 92	[0 1 0]
17 19 23 35 ... 45 81 23 11 23 ... 87 ... 54 24 45 32 ... 56	32 22 76 86 ... 33 18 94 70 67 ... 54 ... 45 67 55 39 ... 48	22 65 11 38 ... 21 94 42 75 30 ... 98 ... 55 32 65 71 ... 92	[1 0 0]
17 19 23 35 ... 45 81 23 11 23 ... 87 ... 54 24 45 32 ... 56	32 22 76 86 ... 33 18 94 70 67 ... 54 ... 45 67 55 39 ... 48	22 65 11 38 ... 21 94 42 75 30 ... 98 ... 55 32 65 71 ... 92	[0 0 1]
...			
17 19 23 35 ... 45 81 23 11 23 ... 87 ... 54 24 45 32 ... 56	32 22 76 86 ... 33 18 94 70 67 ... 54 ... 45 67 55 39 ... 48	22 65 11 38 ... 21 94 42 75 30 ... 98 ... 55 32 65 71 ... 92	[1 0 0]

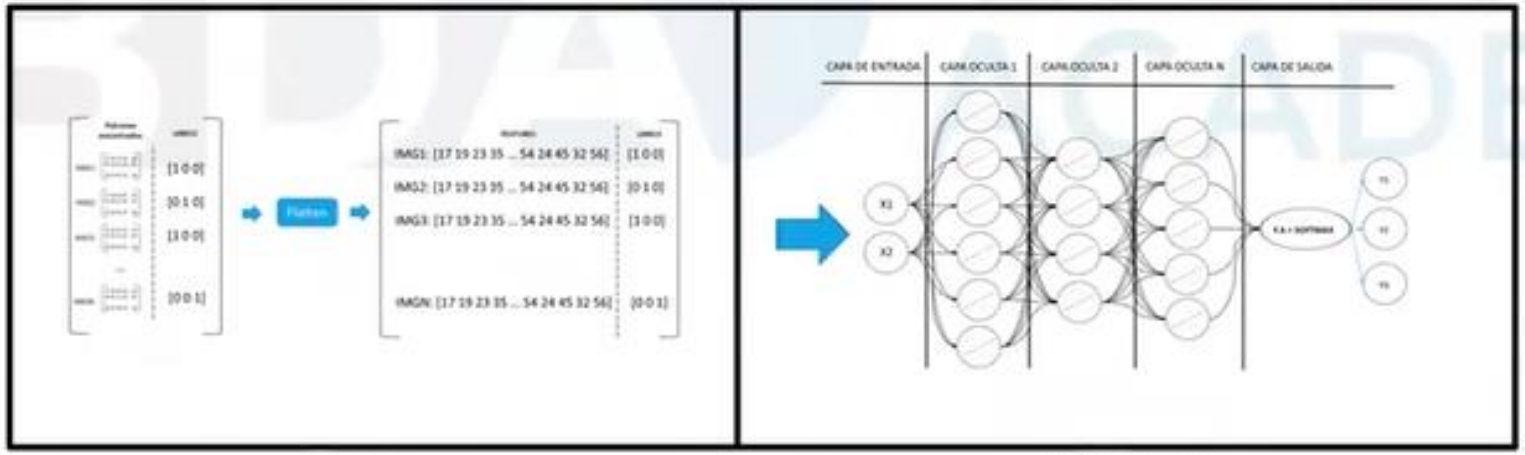
Representación tensorial del Dataset

Arquitectura para Visión Artificial



Feature Engineering

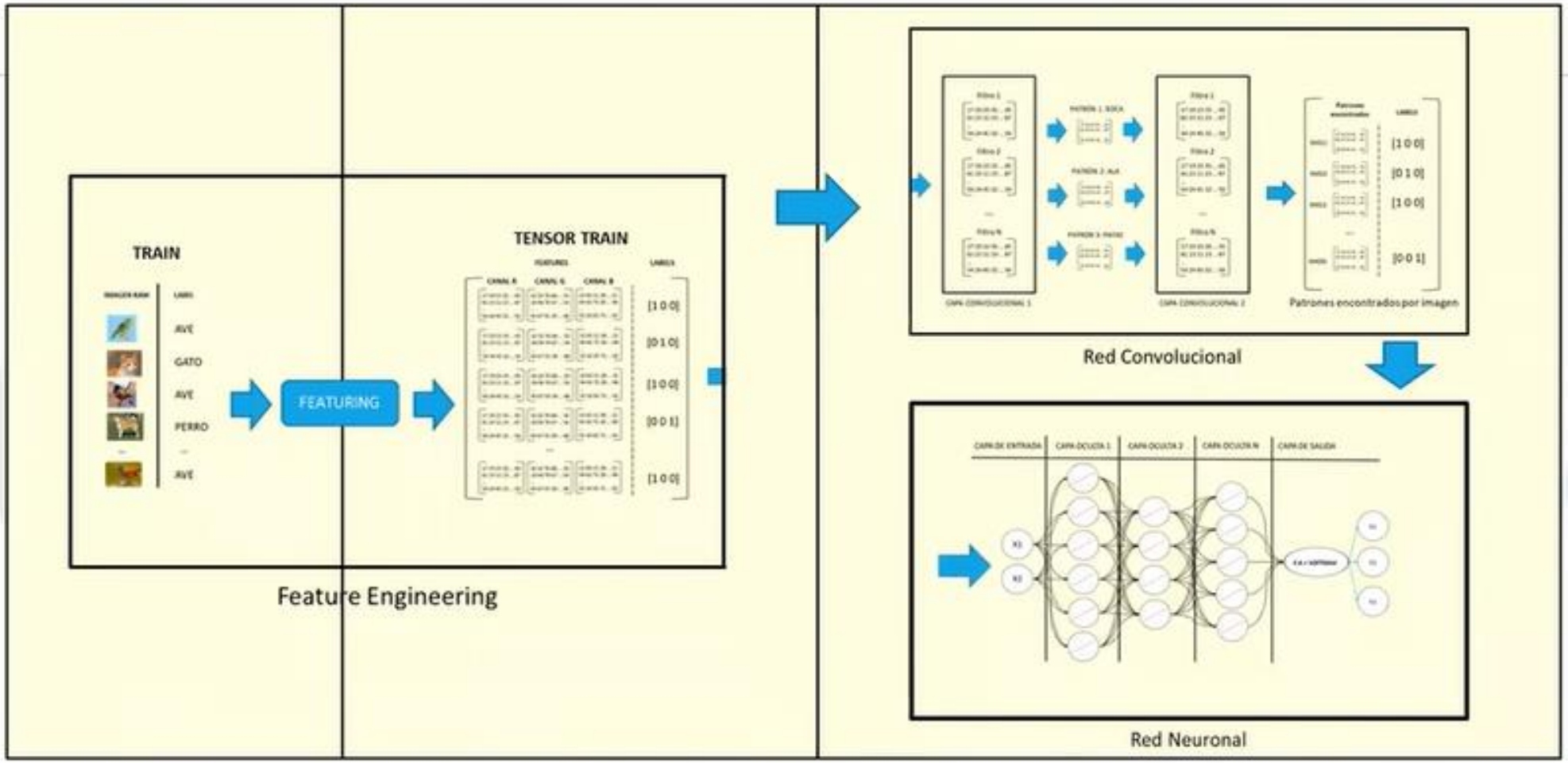
Red Convolucional



Flatten

Red Neuronal

Machine Learning on Datalake

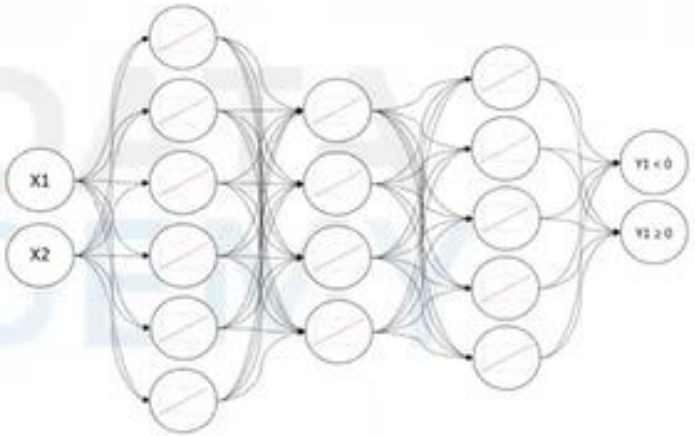


Uso de GPU para procesos matriciales

Ya que la data la tenemos en forma matricial y usamos el álgebra lineal, podemos paralelizar aún más el procesamiento usando GPU

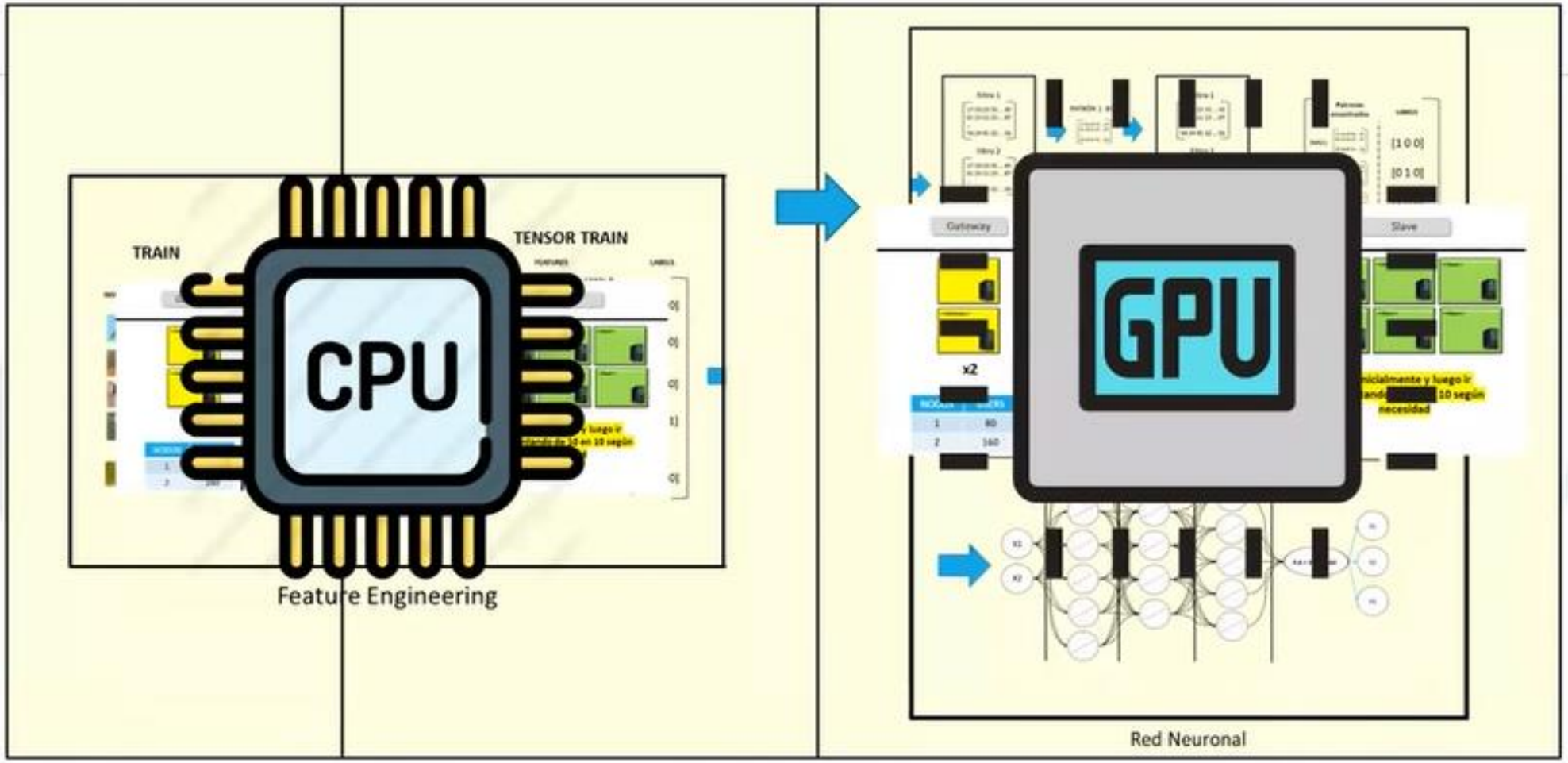
$$X = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \end{array} \begin{array}{c} \begin{array}{ccccc} X_1 & X_2 & X_3 & X_4 & X_5 \end{array} \\ \begin{bmatrix} 0.3 & 0.1 & 0.2 & 0.5 & 0.1 \\ 0.7 & 0.9 & 0.2 & 0.7 & 0.8 \\ 0.2 & 0.6 & 0.2 & 0.8 & 0.6 \\ 0.9 & 0.3 & 0.9 & 0.9 & 0.1 \\ 0.1 & 0.4 & 0.4 & 0.4 & 0.9 \\ 1.0 & 0.8 & 0.8 & 0.2 & 0.7 \\ 0.9 & 0.7 & 0.7 & 0.2 & 0.6 \end{bmatrix} \end{array}$$

$$Y = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \end{array} \begin{array}{c} \begin{array}{c} Y_1 \end{array} \\ \begin{bmatrix} 0.0 \\ 1.0 \\ 1.0 \\ 1.0 \\ 0.0 \\ 1.0 \\ 0.0 \end{bmatrix} \end{array}$$



¡La GPU es especialista en realizar cálculos de productos matriciales!, **factor de aumento en velocidad de entre x2 a x10**

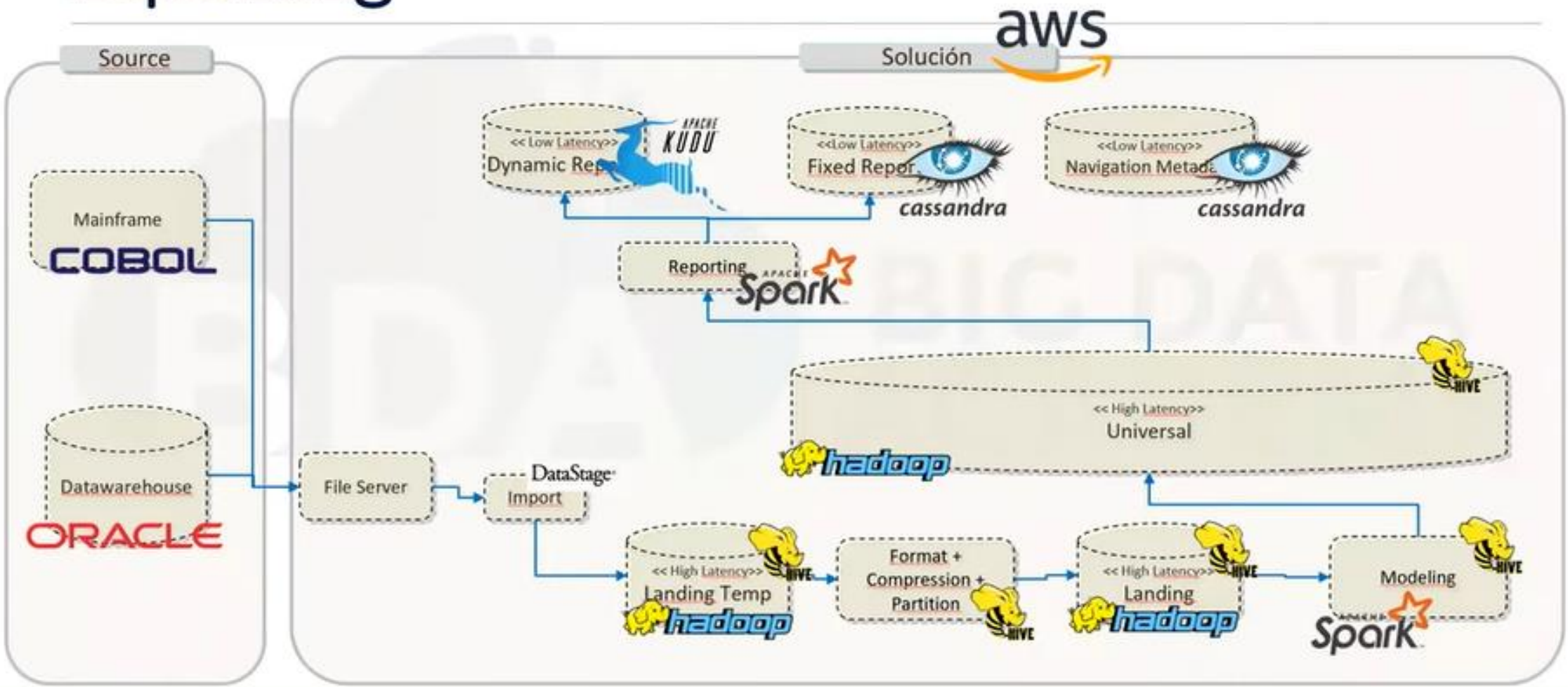
Infraestructura para Machine Learning on Datalake



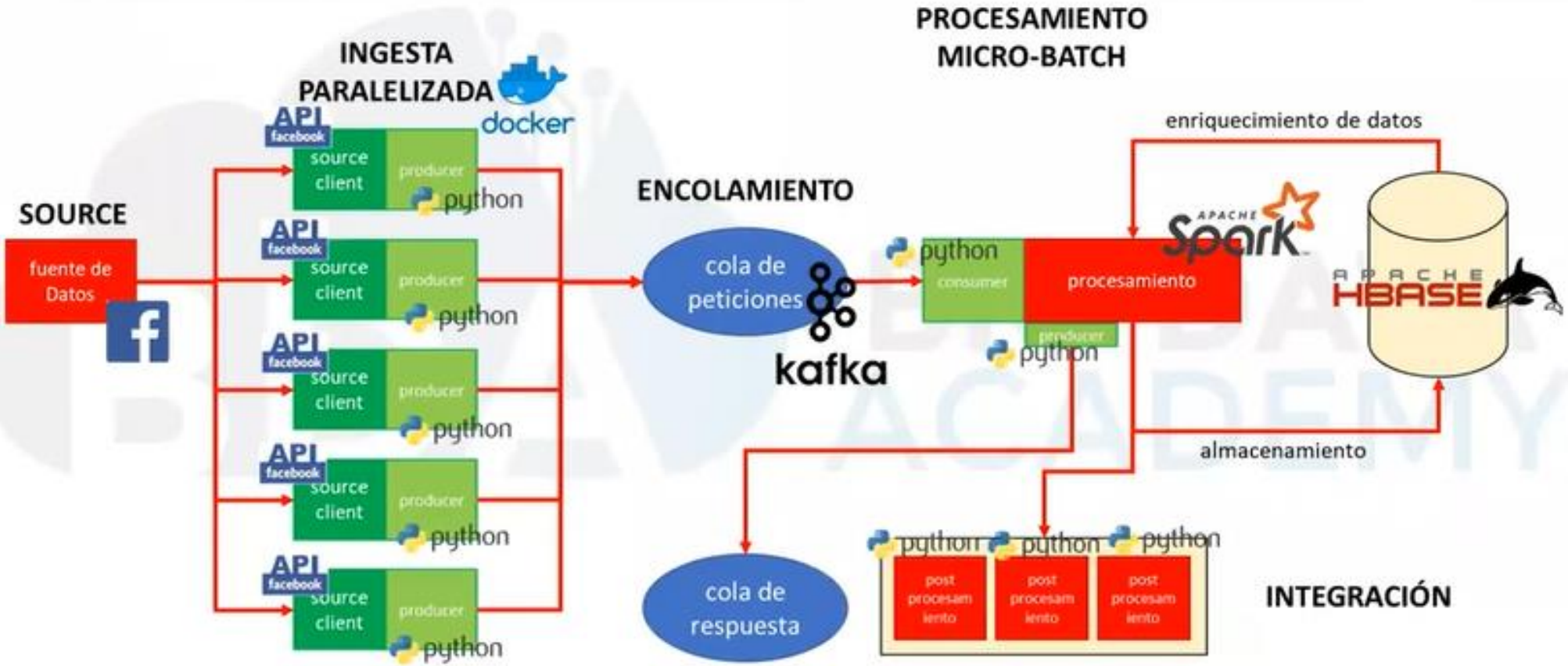
Vista de patrones de diseño [ETL, Reporting, Real-Time]

HAGA CLIC PARA AGREGAR TEXTO

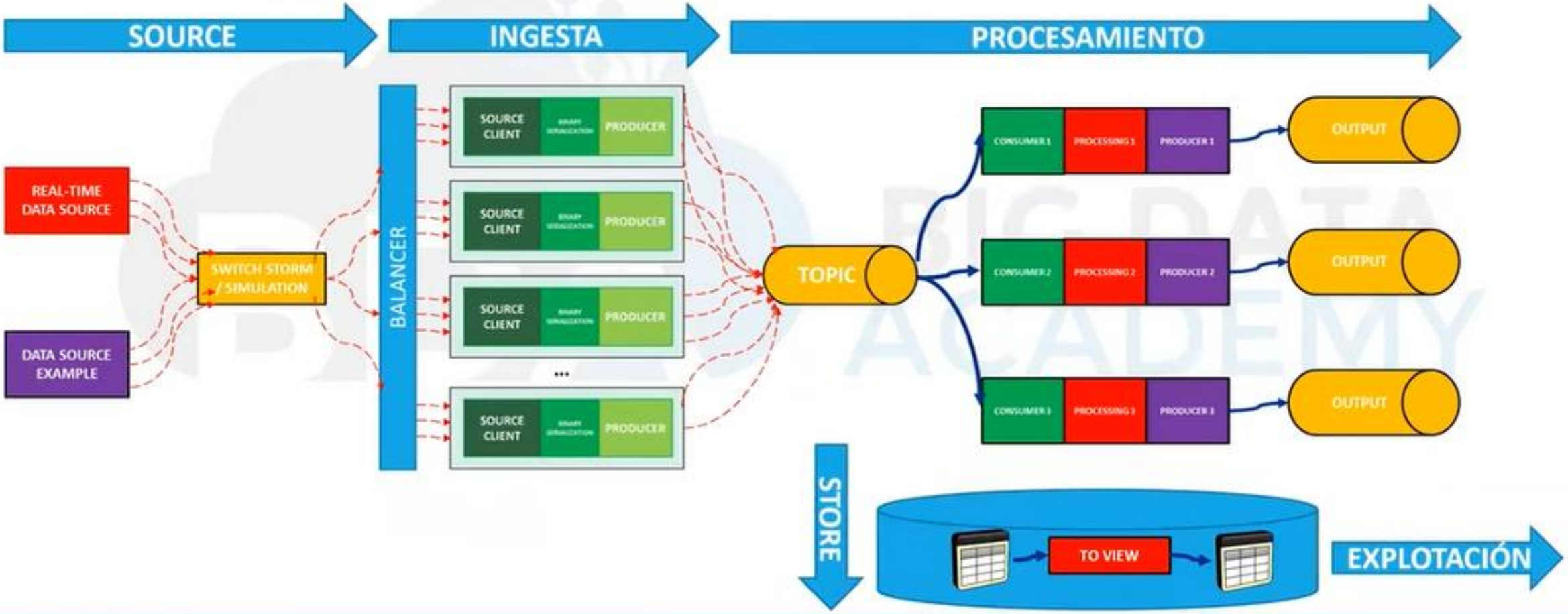
Patrón de procesamiento batch aplicado a reporting



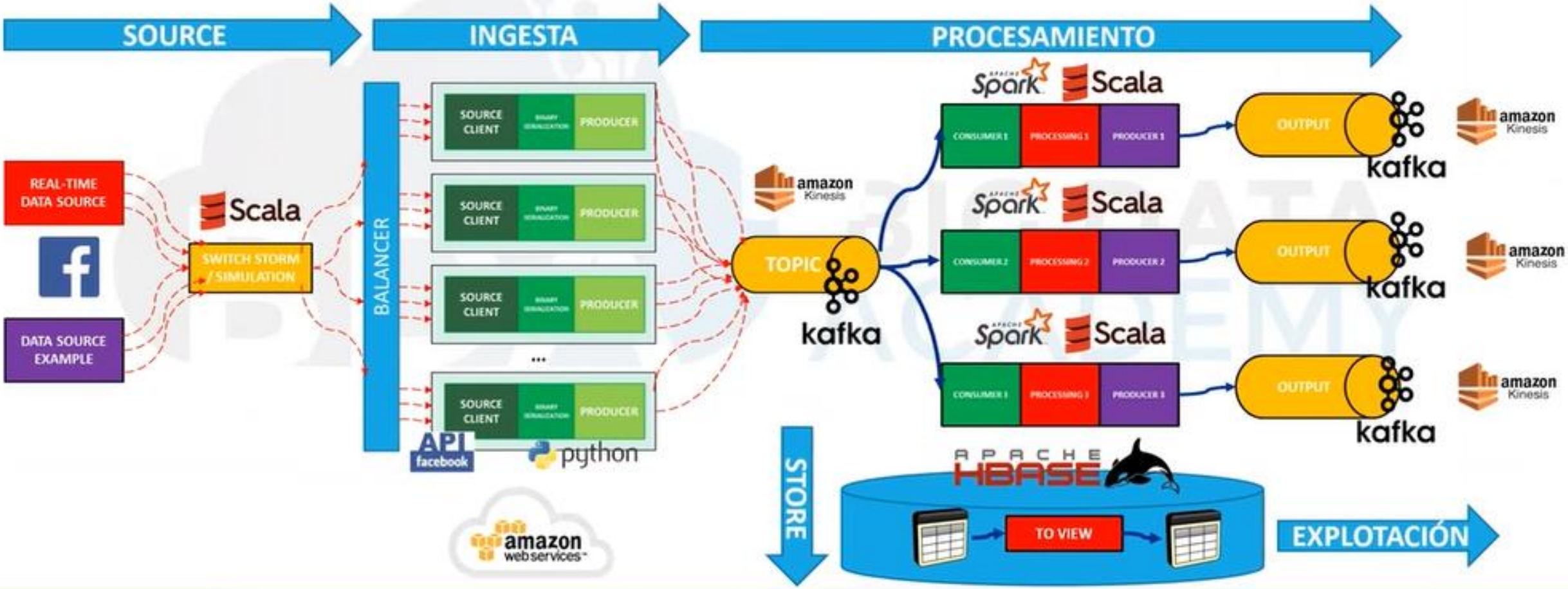
Patrón de procesamiento en real time aplicado a redes sociales



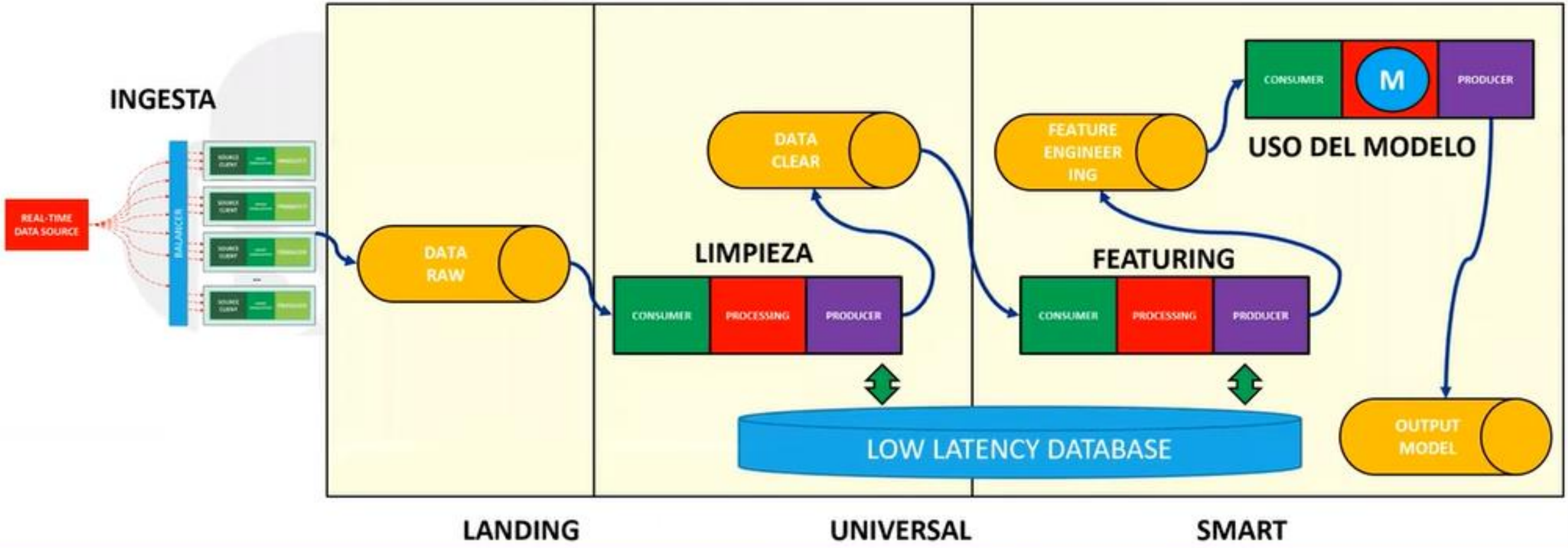
Arquitectura estándar extendida para procesamiento en Real-Time



Arquitectura estándar extendida para procesamiento en Real-Time



Machine Learning on Datalake para explotación en Real-Time

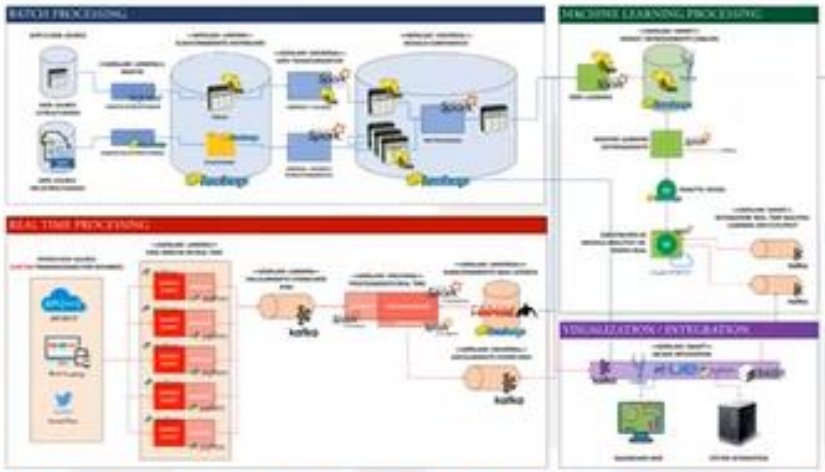


Conclusiones

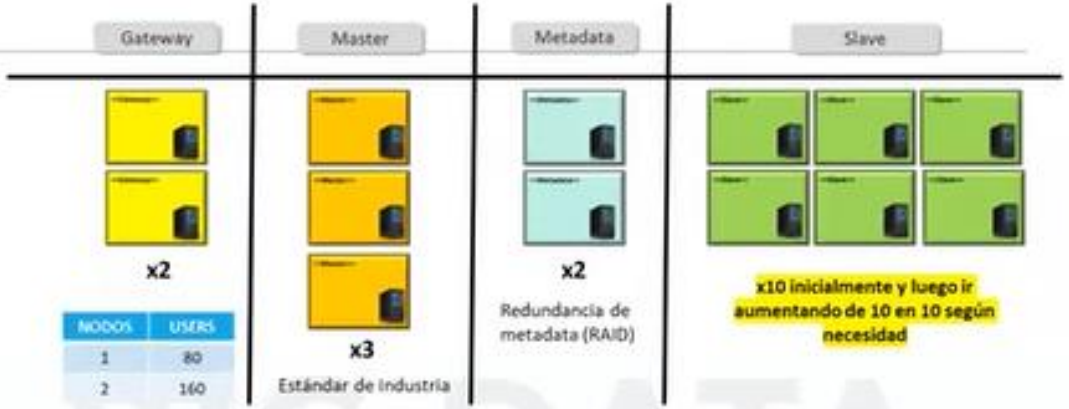
HAGA CLIC PARA AGREGAR TEXTO

Próximos pasos

Arquitecturas y patrones

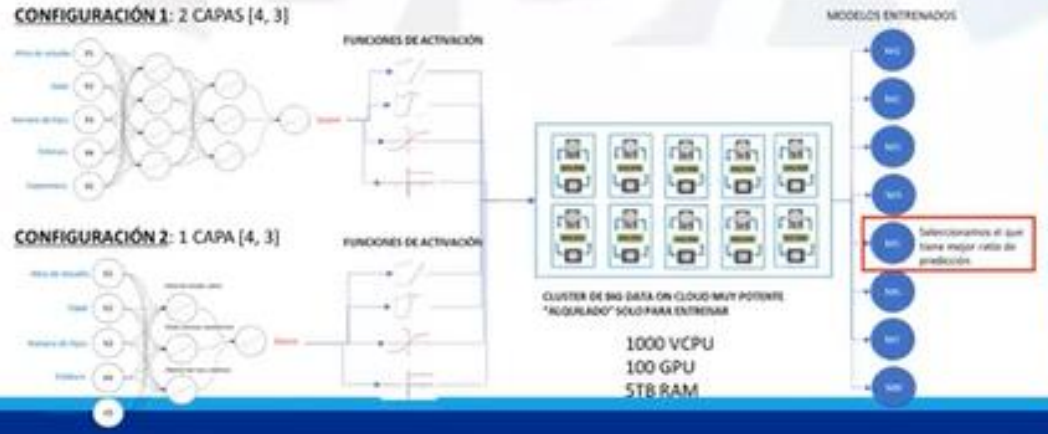


Infraestructura Cloud y On-Premise



NODOS	USERS
1	80
2	160

Machine Learning Escalable



Arquetipos de código

```
def formatData(register):
    registerFormatted = []
    try:
        registerValue = register[1]
        registerFormatted = json.loads(registerValue)
    except Exception as e:
        registerFormatted = []
    return registerFormatted

def enrichmentData(register):
    dateTime = datetime.datetime.now()
    register['FECHA_TRANSACCION'] = str(dateTime.date())
    register['HORA_TRANSACCION'] = str(dateTime.time())
```