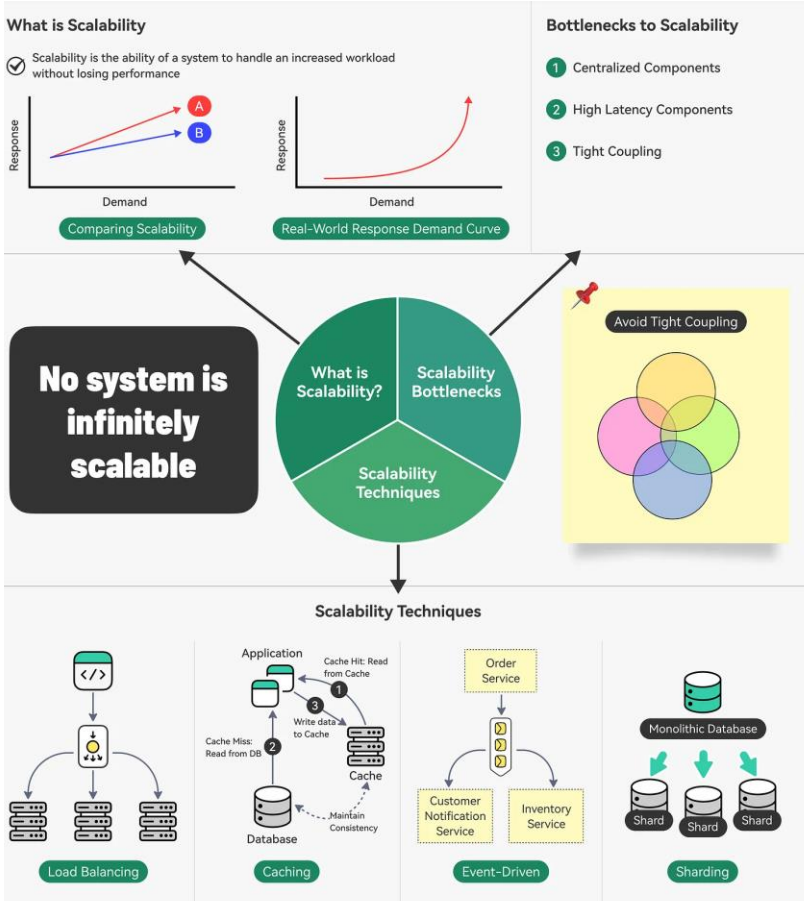
BIG DATA

- **Material Elaborado por Profesor: Sergio Gevaschnaider**

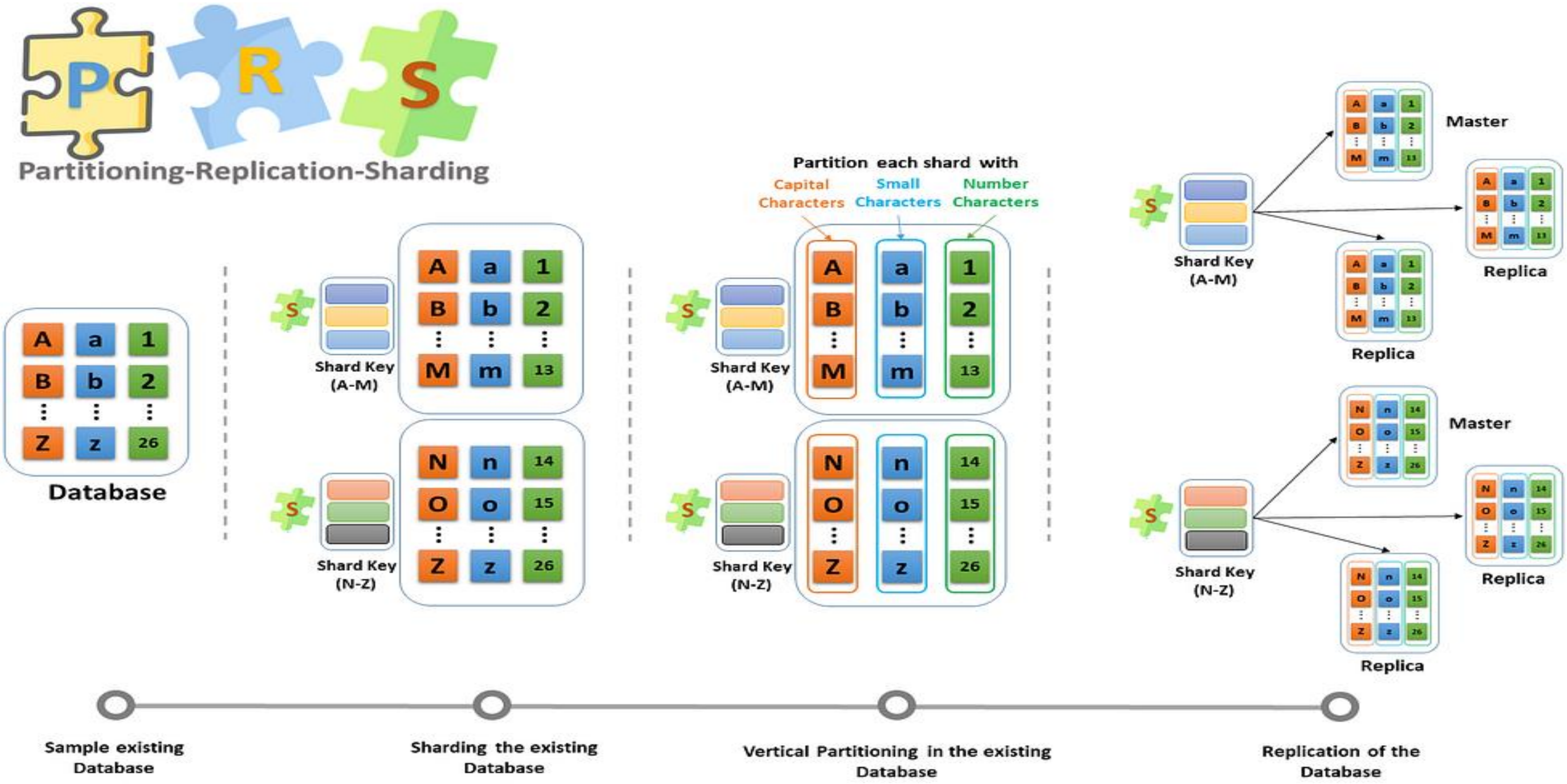
Photo by tian ci on Unsplash

Problemas de escalabilidad



Factor	Internet	Escalabilidad	Big Data
Inicio	Años 60-70: Redes ARPANET.	Hardware tradicional con servidores únicos.	Bases de datos relacionales y almacenamiento en disco.
Crecimiento	Años 90: Web 1.0 y comercio electrónico.	Surgimiento de balanceadores de carga y bases de datos distribuidas.	Aumento de datos no estructurados y necesidad de nuevos modelos de almacenamiento.
Explosión	Años 2000: Redes sociales, Web 2.0.	Se popularizan arquitecturas de microservicios y balanceo de carga global.	Big Data toma fuerza con Hadoop, Spark y NoSQL.
Presente y futuro	Web 3.0, IoT, Inteligencia Artificial en la nube.	Edge computing y arquitecturas serverless para manejar tráfico global.	DataOps, procesamiento en tiempo real, integración con AI.

Problemas con el sharding



Principios de un sistema de Big Data

1. Robustez y Tolerancia a Fallos
   - Inmutabilidad
   - Recomputation

2. Lecturas y Actualizaciones de Baja Latencia
   - Latencia de Lectura
   - Latencia de Actualización

3. Escalabilidad
   - Escalabilidad Horizontal
   - Arquitectura Lambda

4. Generalización
   - Sistema Generalizado

5. Extensibilidad
   - Sistema Extensible
   - Migración de Datos

Principios de un sistema de Big Data

6. Consultas Ad Hoc

   - Consulta Ad Hoc

7. Mantenimiento Mínimo

   - Mantenimiento

   - Complejidad de Implementación

8. Depurabilidad

   - Depurabilidad

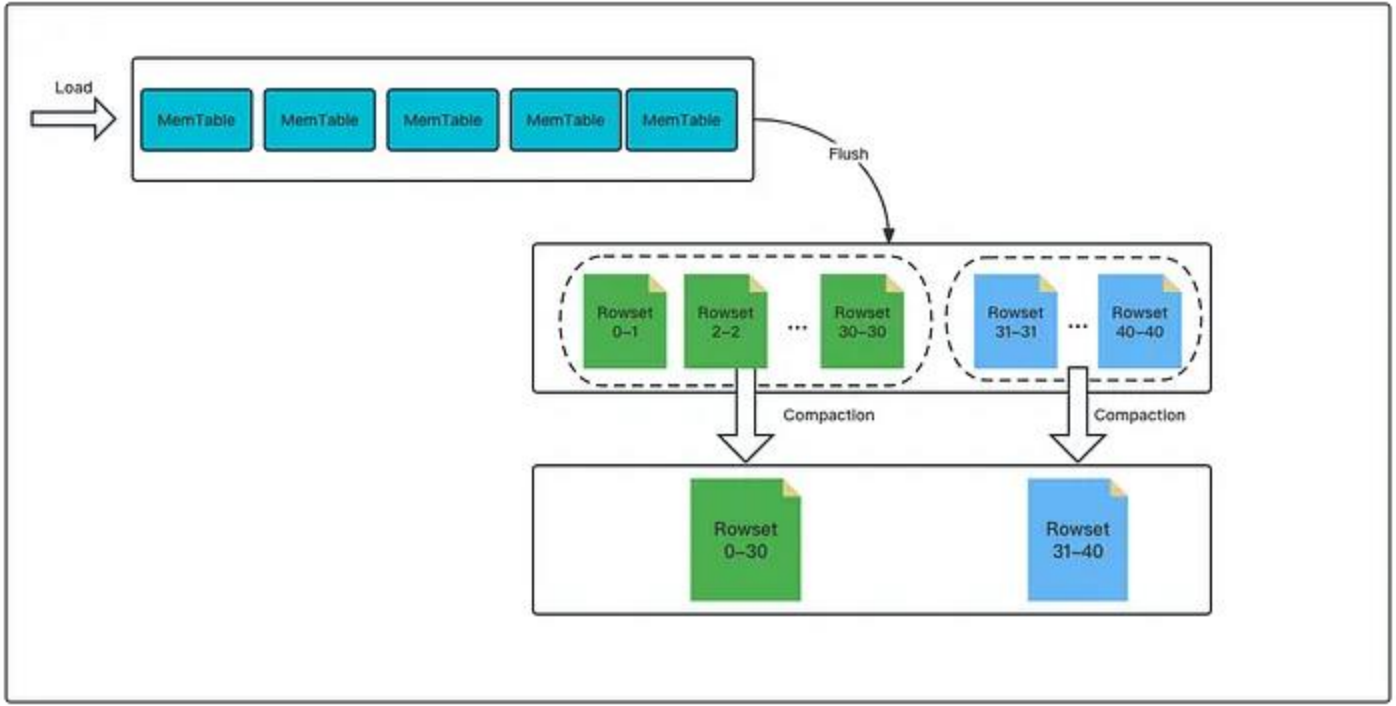
   - Programación Funcional

Programación funcional y Apache Spark

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, avg

spark = SparkSession.builder.appName("Ejemplo").getOrCreate()

# Cargar un DataFrame
df = spark.read.json("datos.json")

# Definir funciones para cada transformación
def filter_by_age(df):
  return df.filter(col("edad") > 20)

def select_columns(df):
  return df.select("nombre", "edad")

def group_by_name(df):
  return df.groupBy("nombre")

def calculate_average(df):
  return df.agg({"edad": "avg"})

# Aplicar las funciones de forma funcional
resultado = calculate_average(group_by_name(select_columns(filter_by_age(df))))

resultado.show()
```

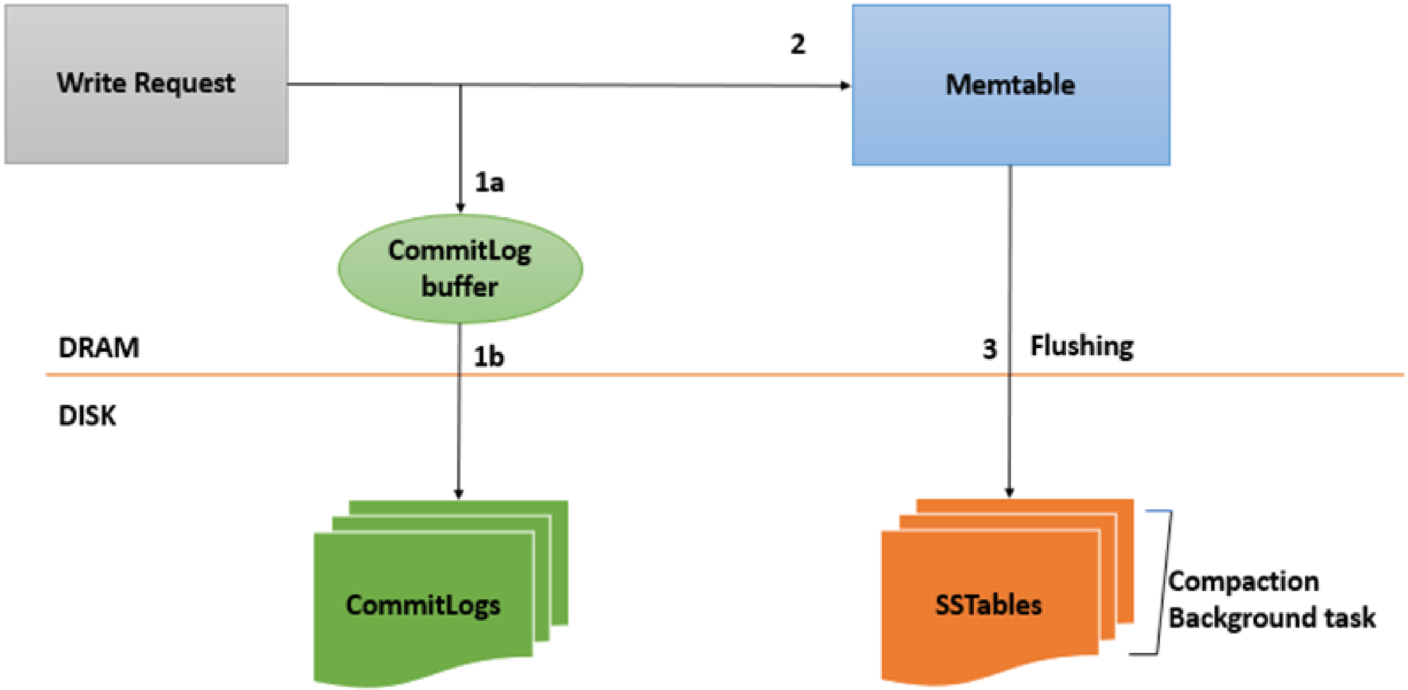
Arquitectura incremental

Necesidad y problemas de la compactación de los datos

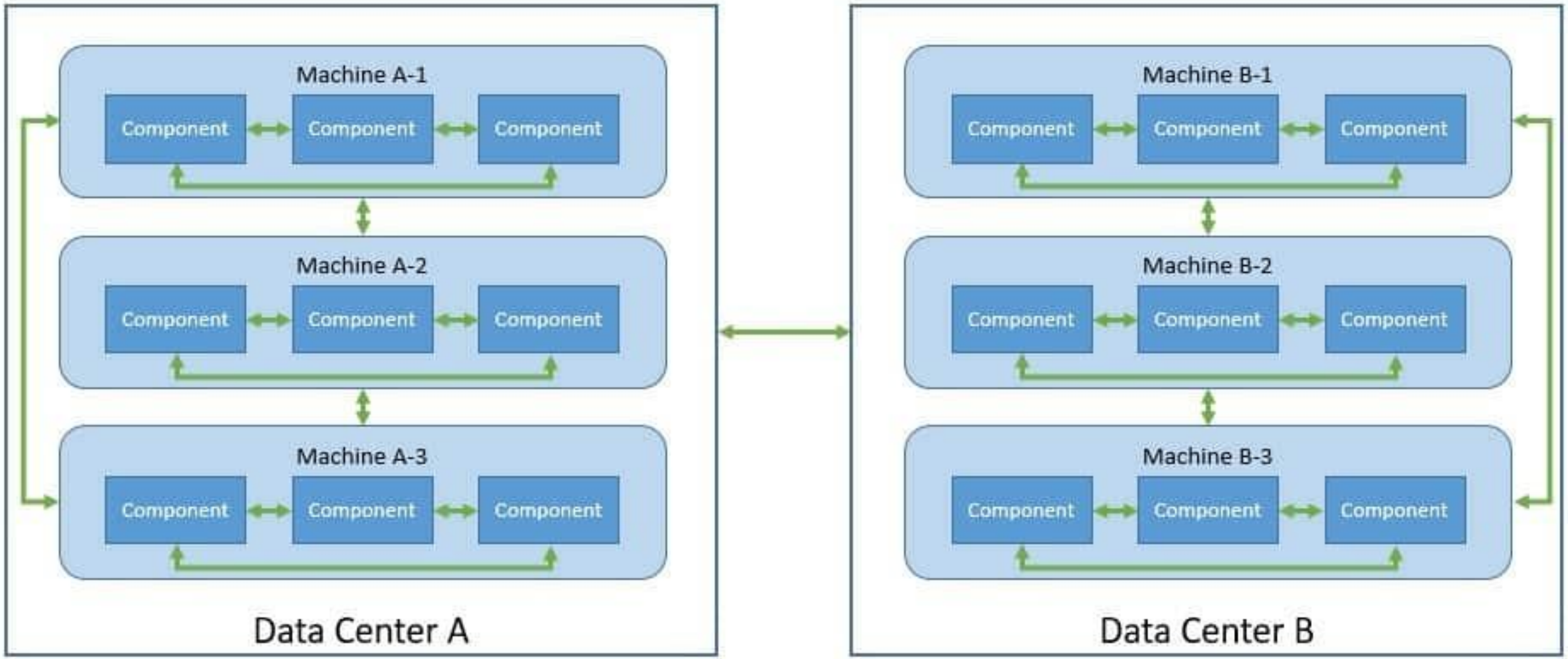


Arquitectura incremental

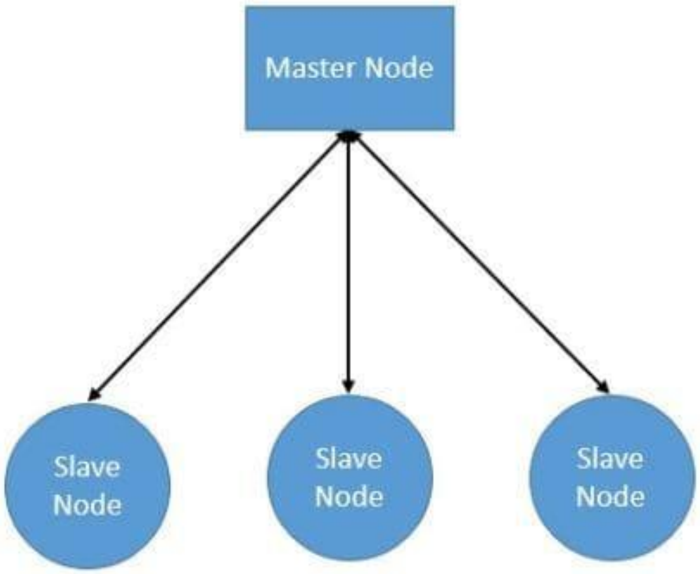
Necesidad y problemas de la compactación de los datos: Cassandra



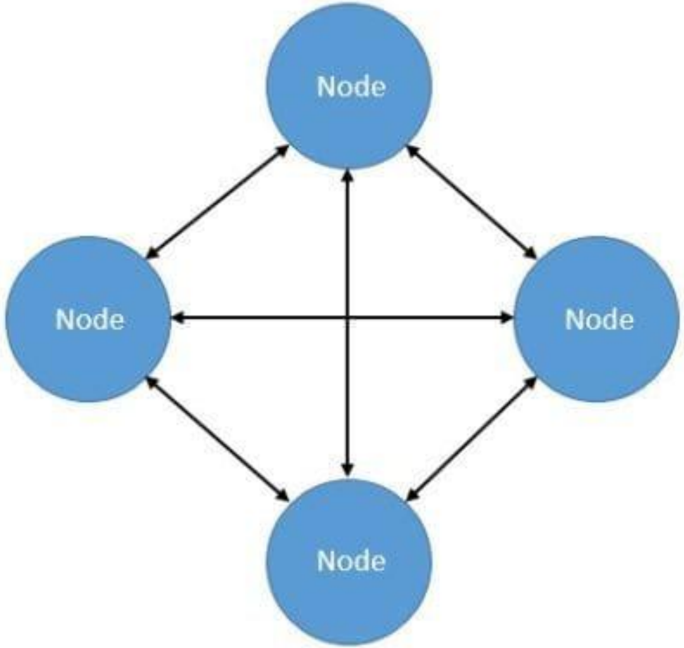
Sistemas distribuidos y CAP



Sistemas distribuidos y CAP

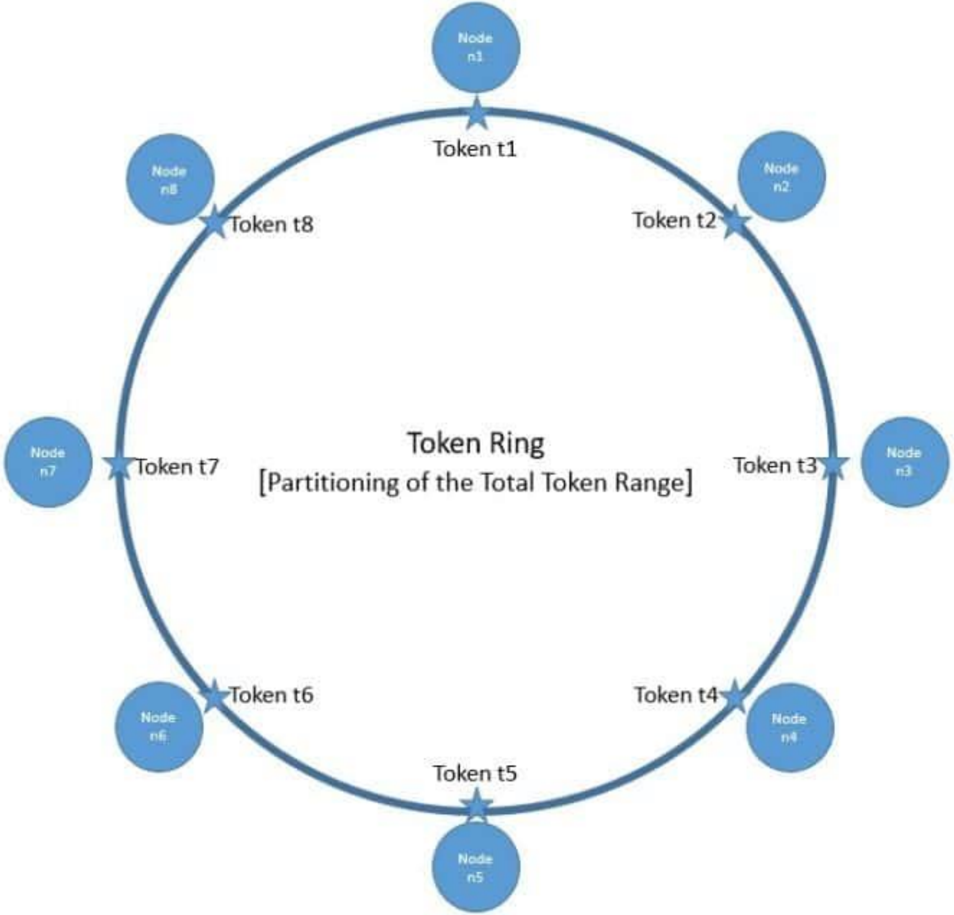


Master-Slave Model



Peer-to-Peer Model

Sistemas distribuidos y CAP: Cassandra

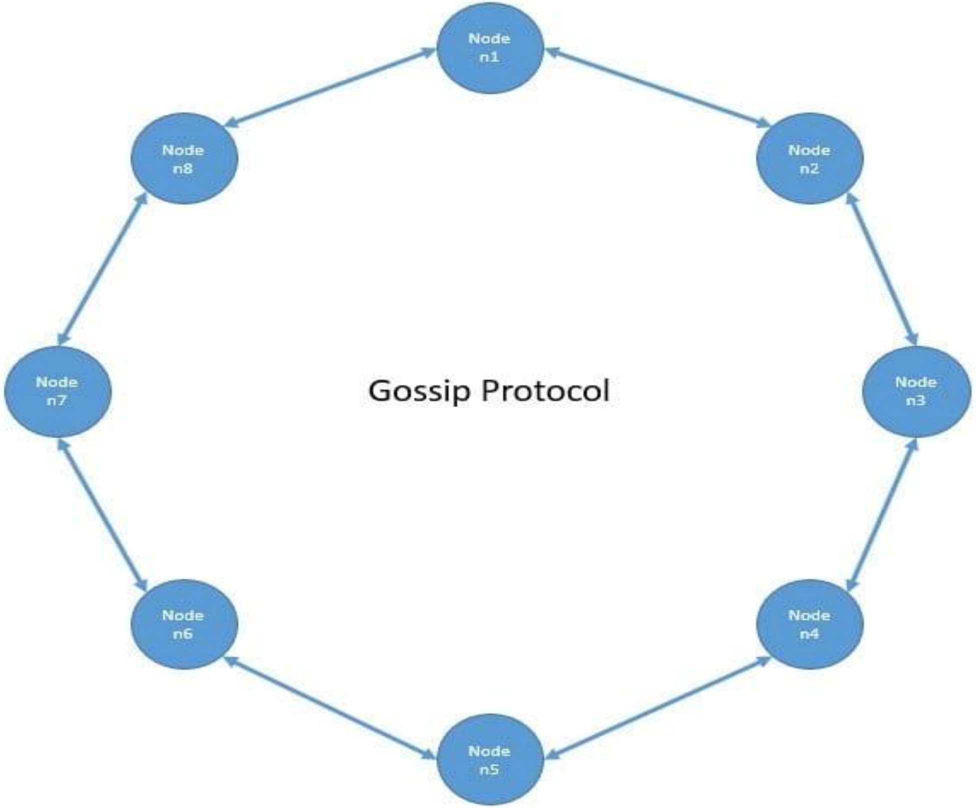


For RF = 2:
- Range(t1, t2] -> {n2, n3}
- Range(t2, t3] -> {n3, n4}

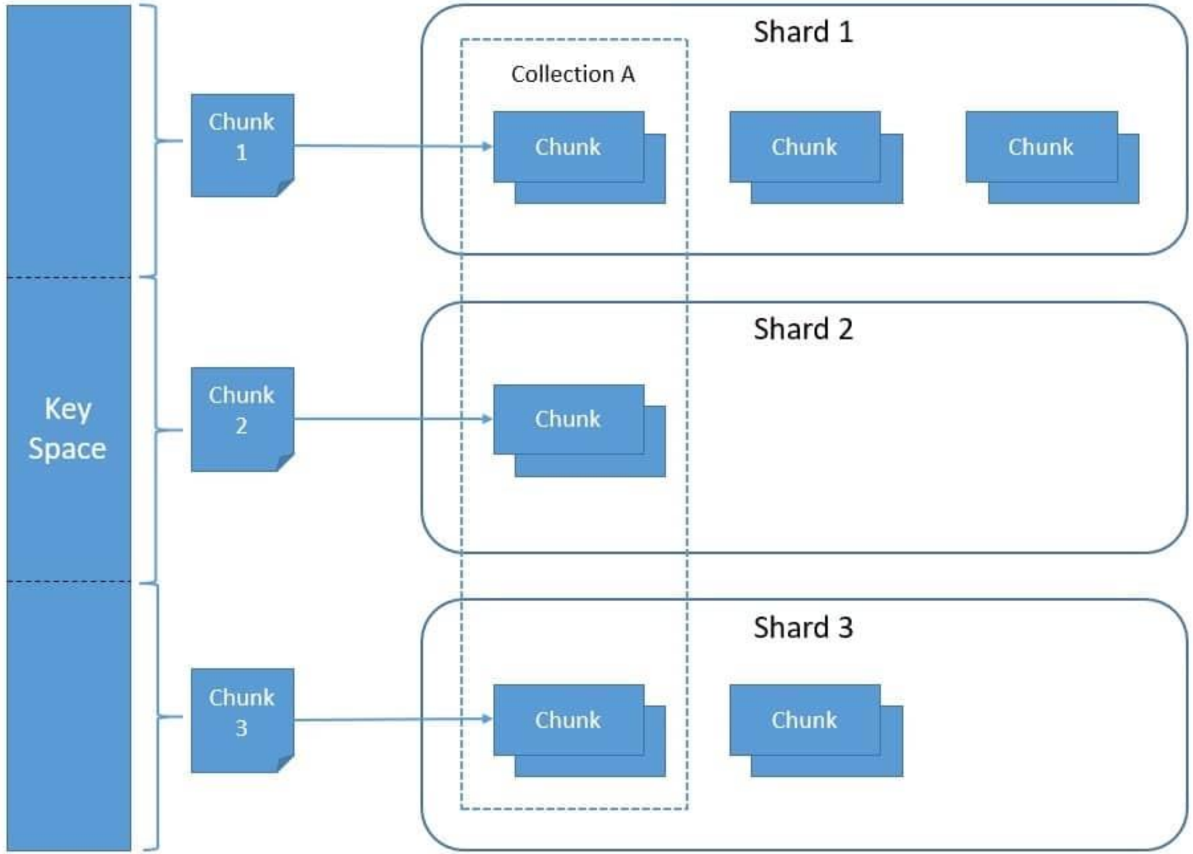
For RF = 3:
- Range(t1, t2] -> {n2, n3, n4}
- Range(t2, t3] -> {n3, n4, n5}

For RF = 4:
- Range(t1, t2] -> {n2, n3, n4, n5}
- Range(t2, t3] -> {n3, n4, n5, n6}

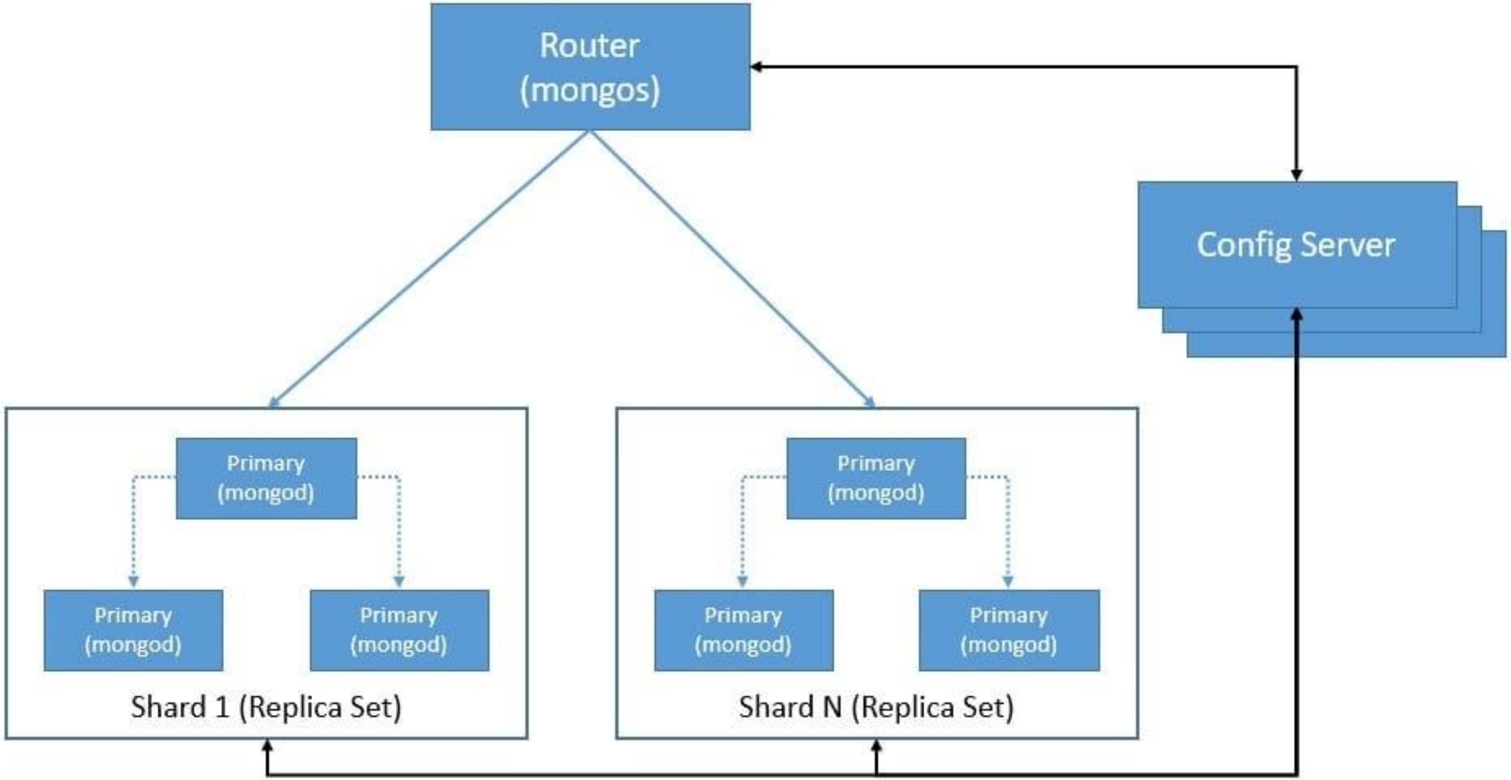
Sistemas distribuidos y CAP: Cassandra



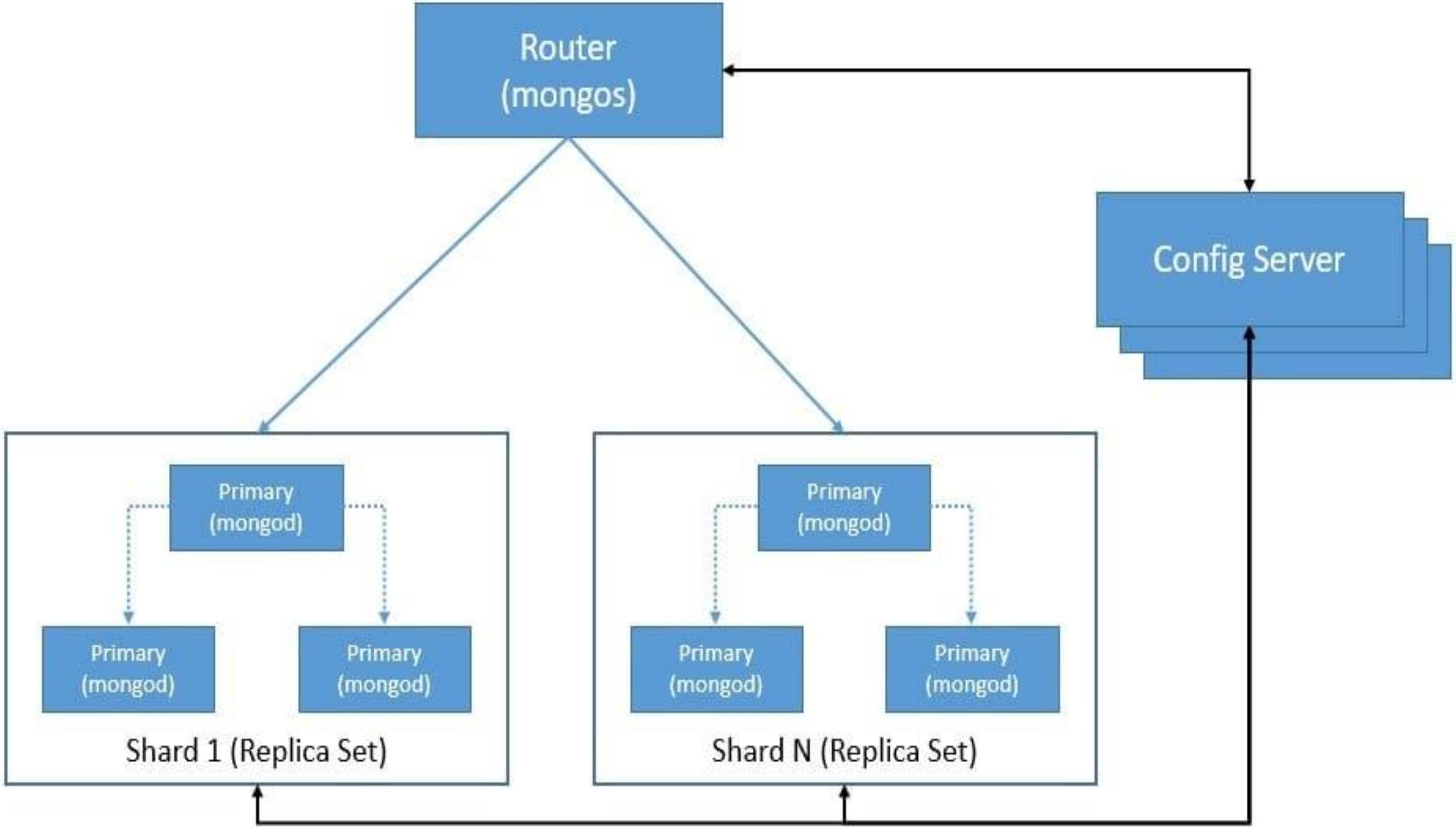
Sistemas distribuidos y CAP: Mongo DB



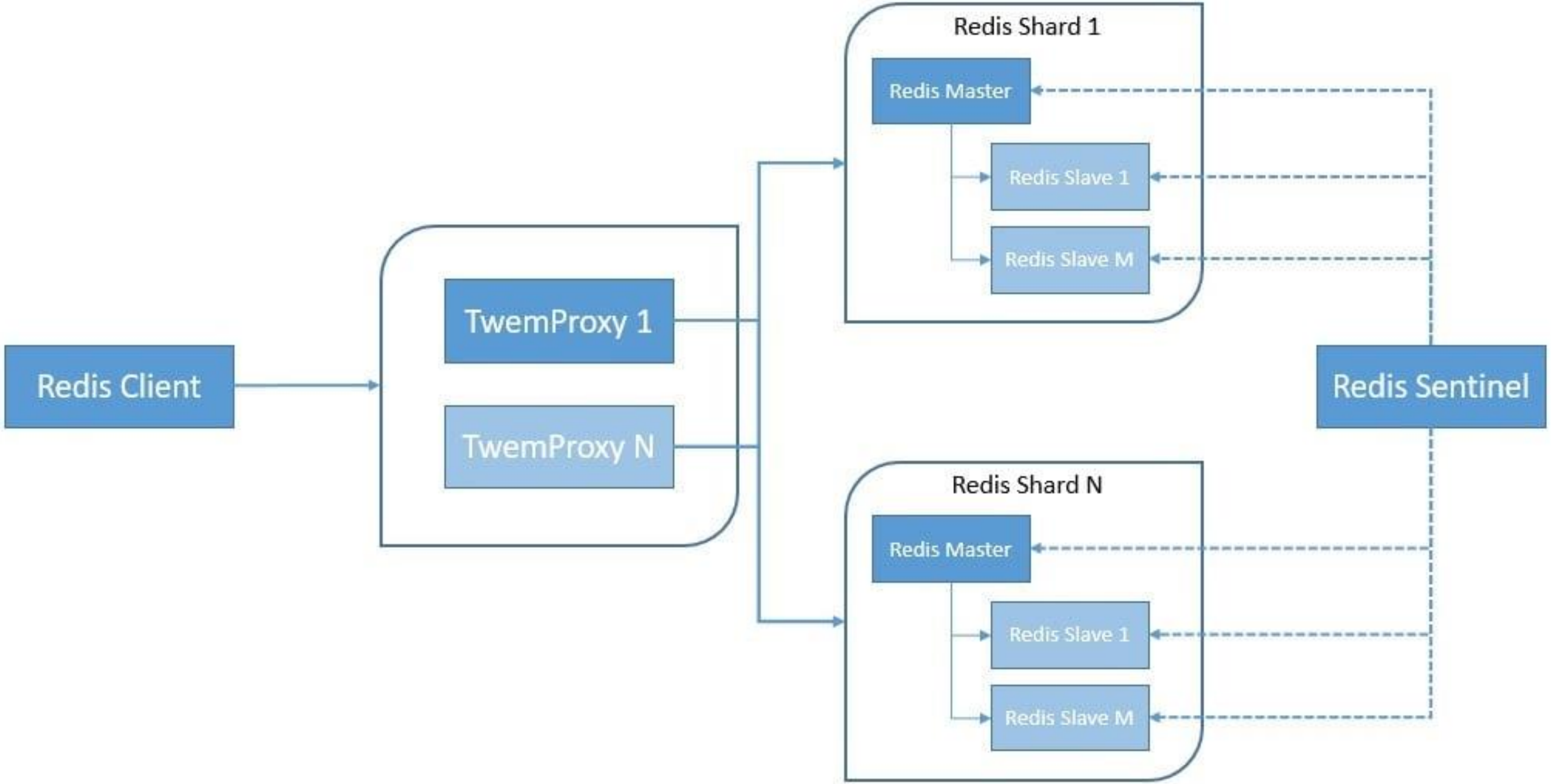
Sistemas distribuidos y Mongo DB



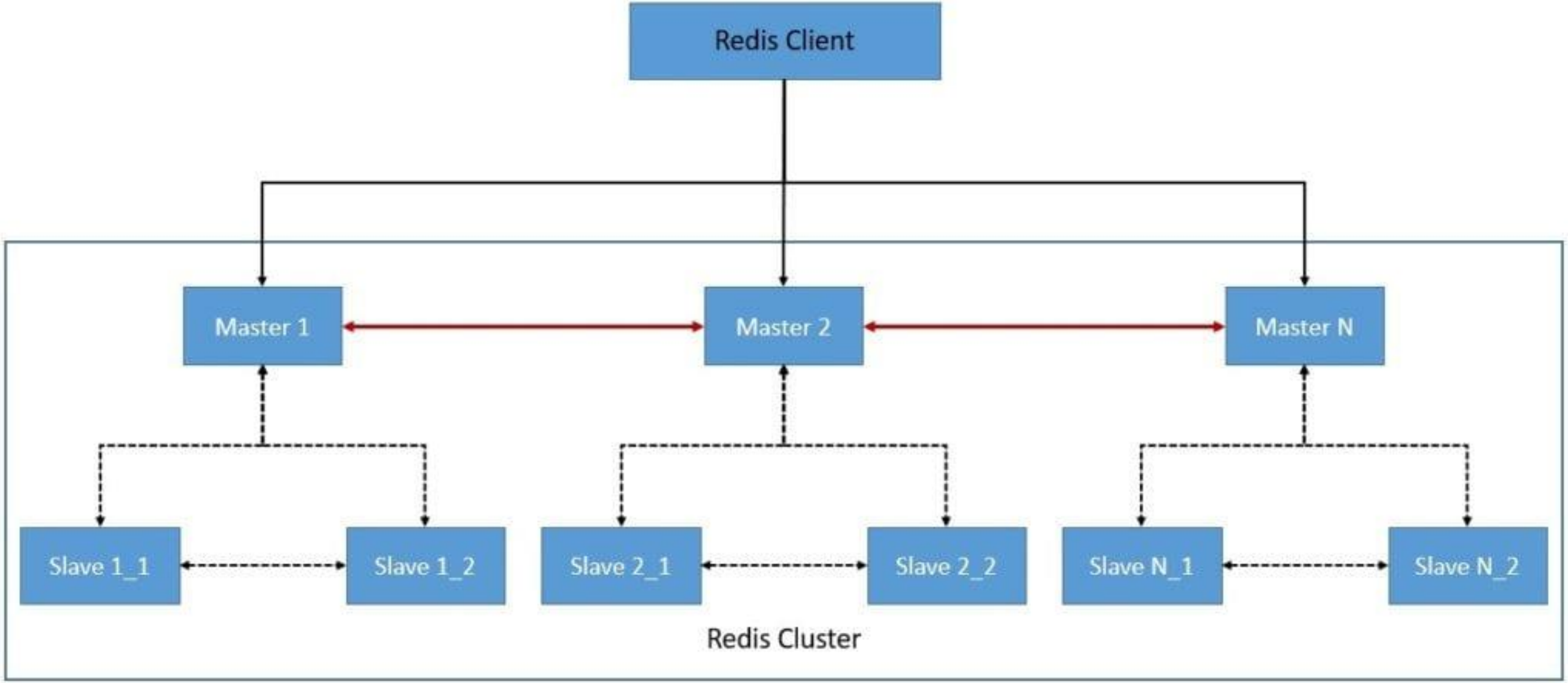
Sistemas distribuidos y Mongo DB: Cassandra



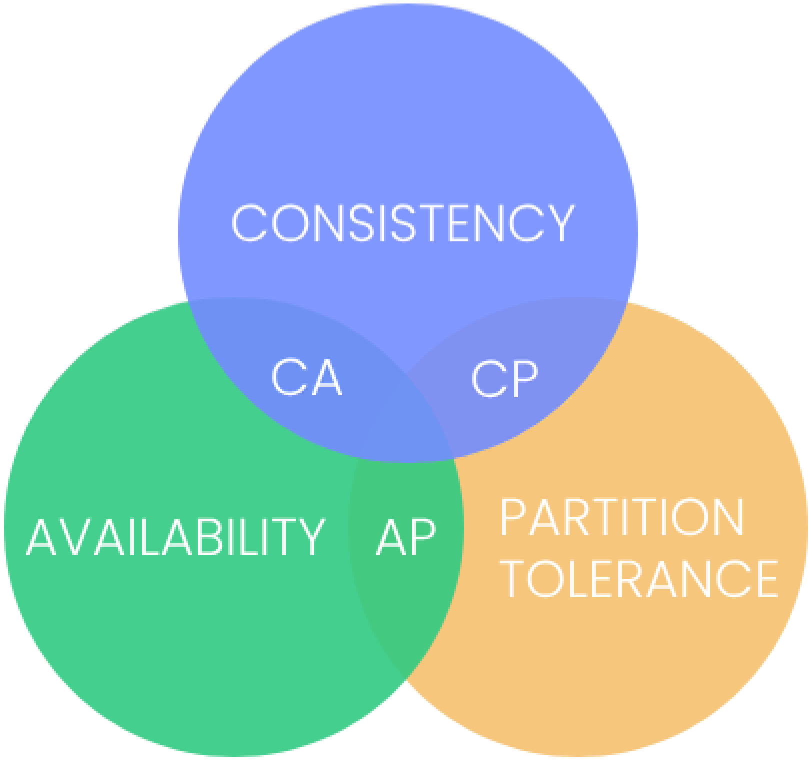
Sistemas distribuidos y Redis



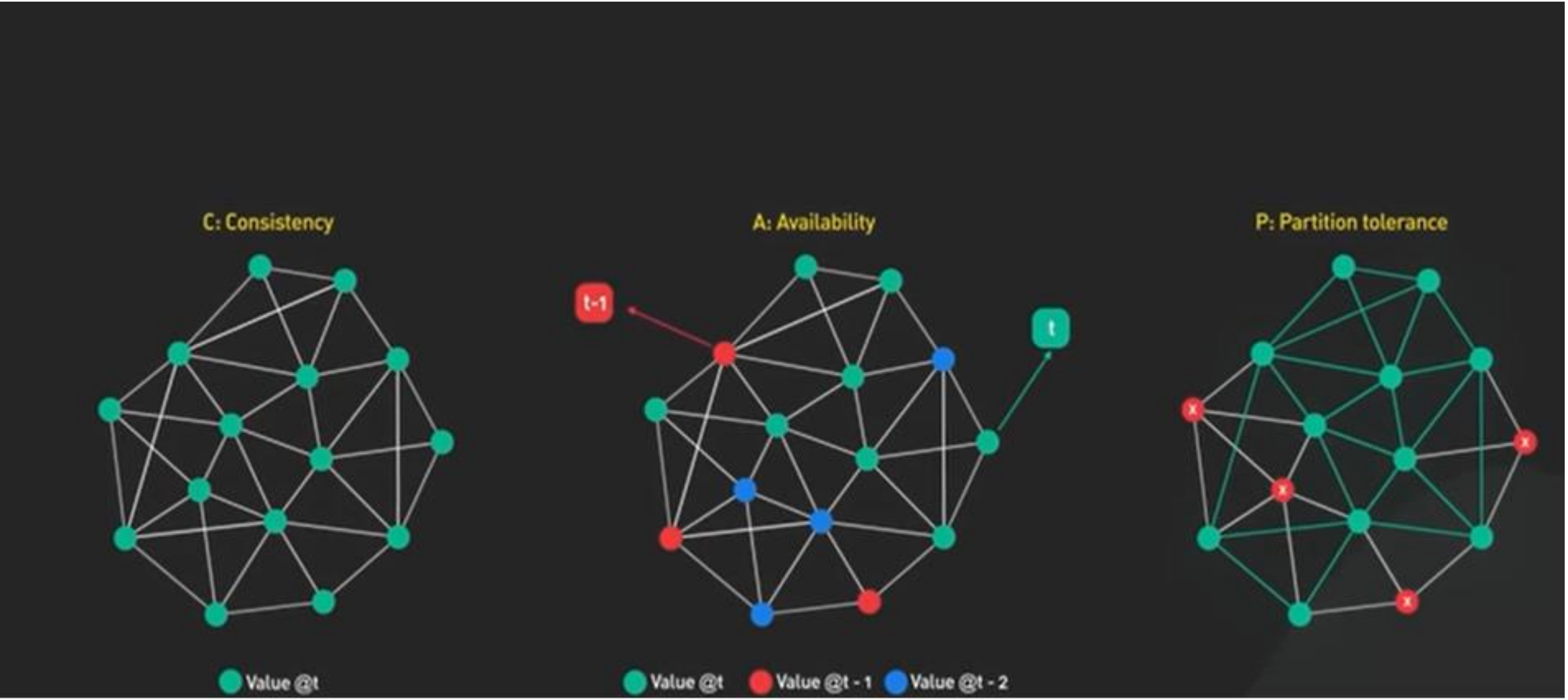
Sistemas distribuidos y Mongo DB: Redis



TEOREMA CAP

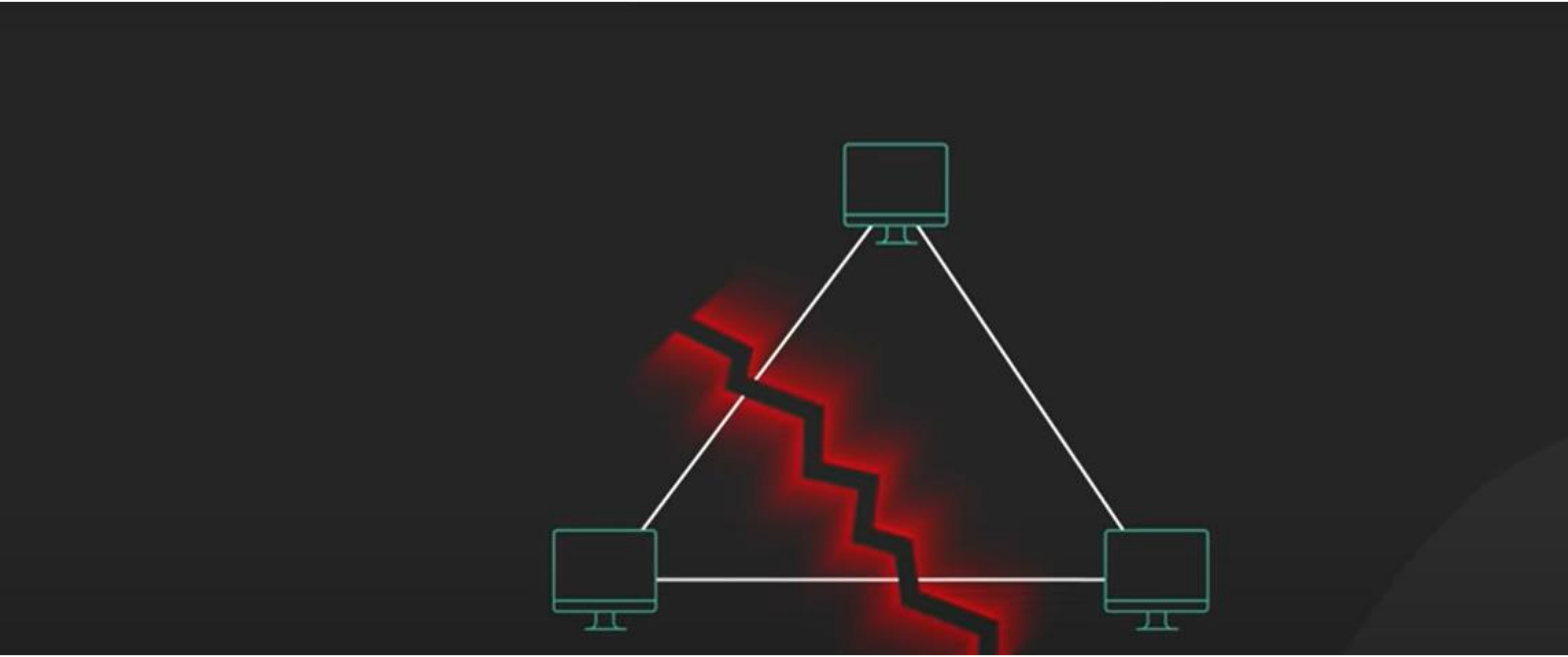


TEOREMA CAP

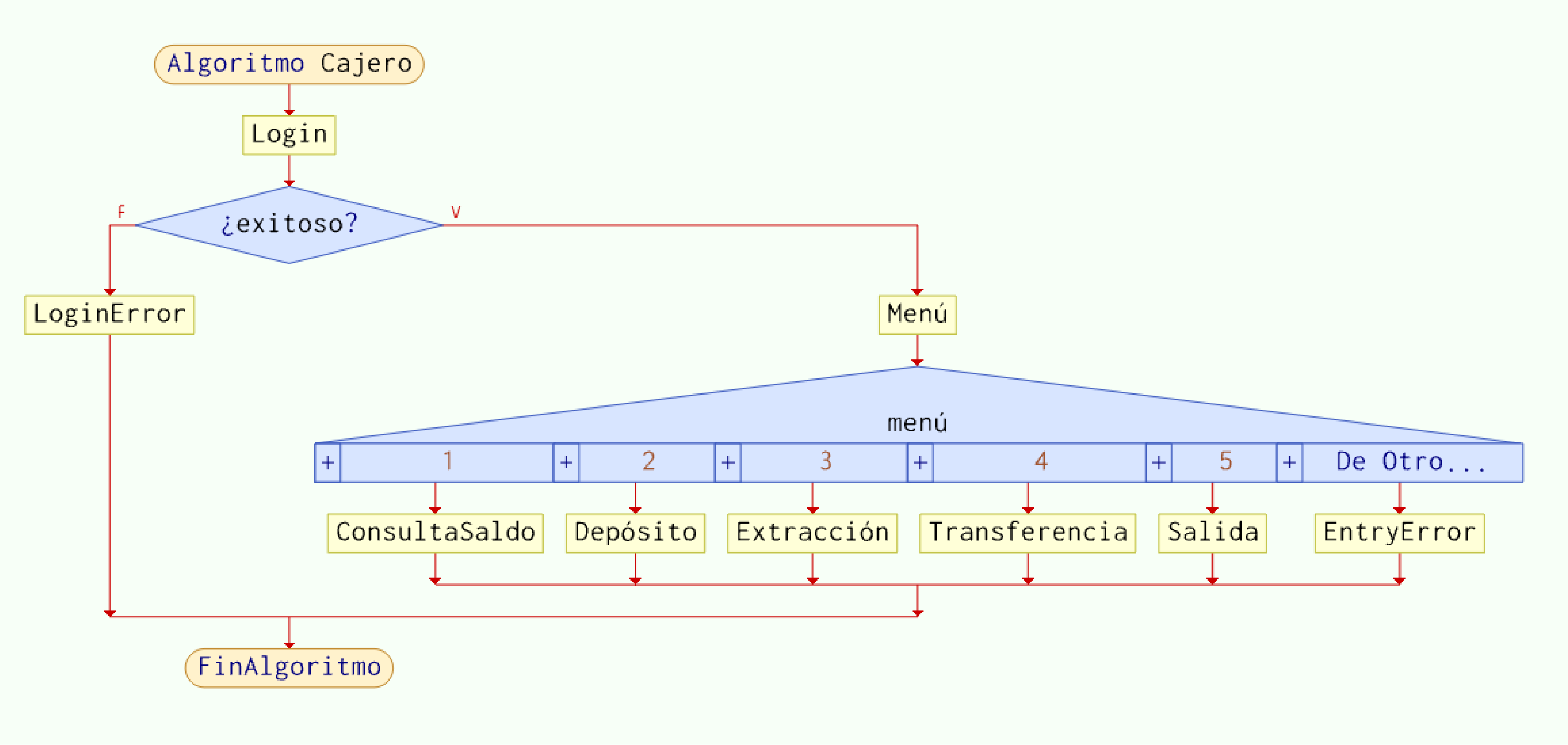


C: Consistency
A: Availability
P: Partition tolerance
t-1
t
x
Value @t
Value @t
Value @t - 1
Value @t - 2

TEOREMA CAP

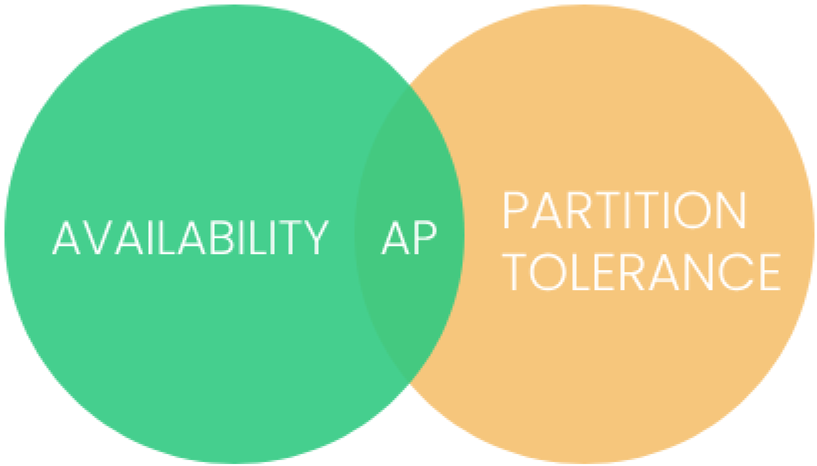


TEOREMA CAP

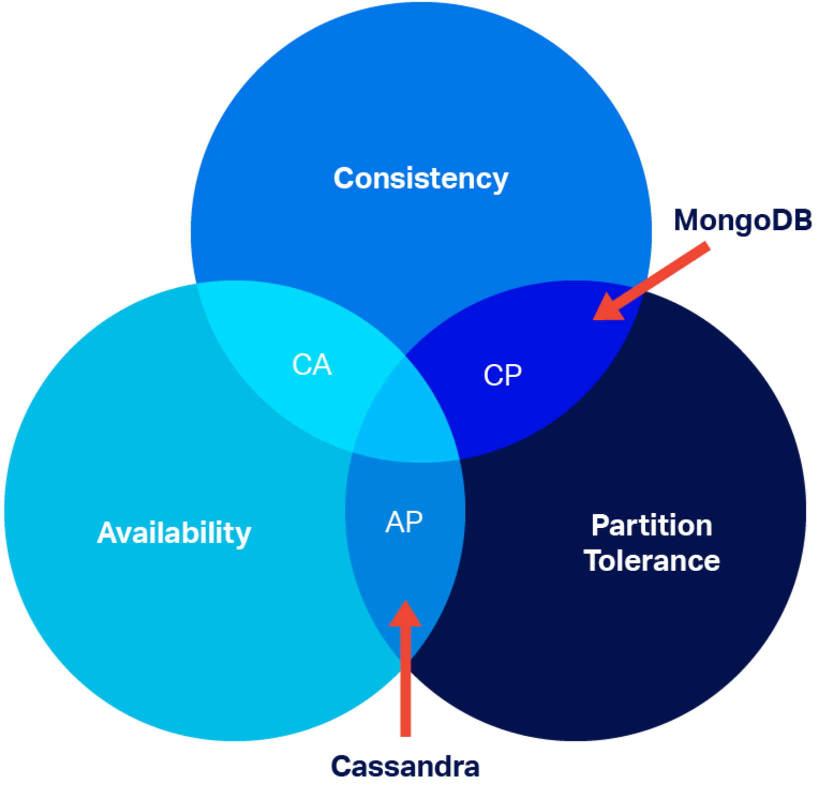


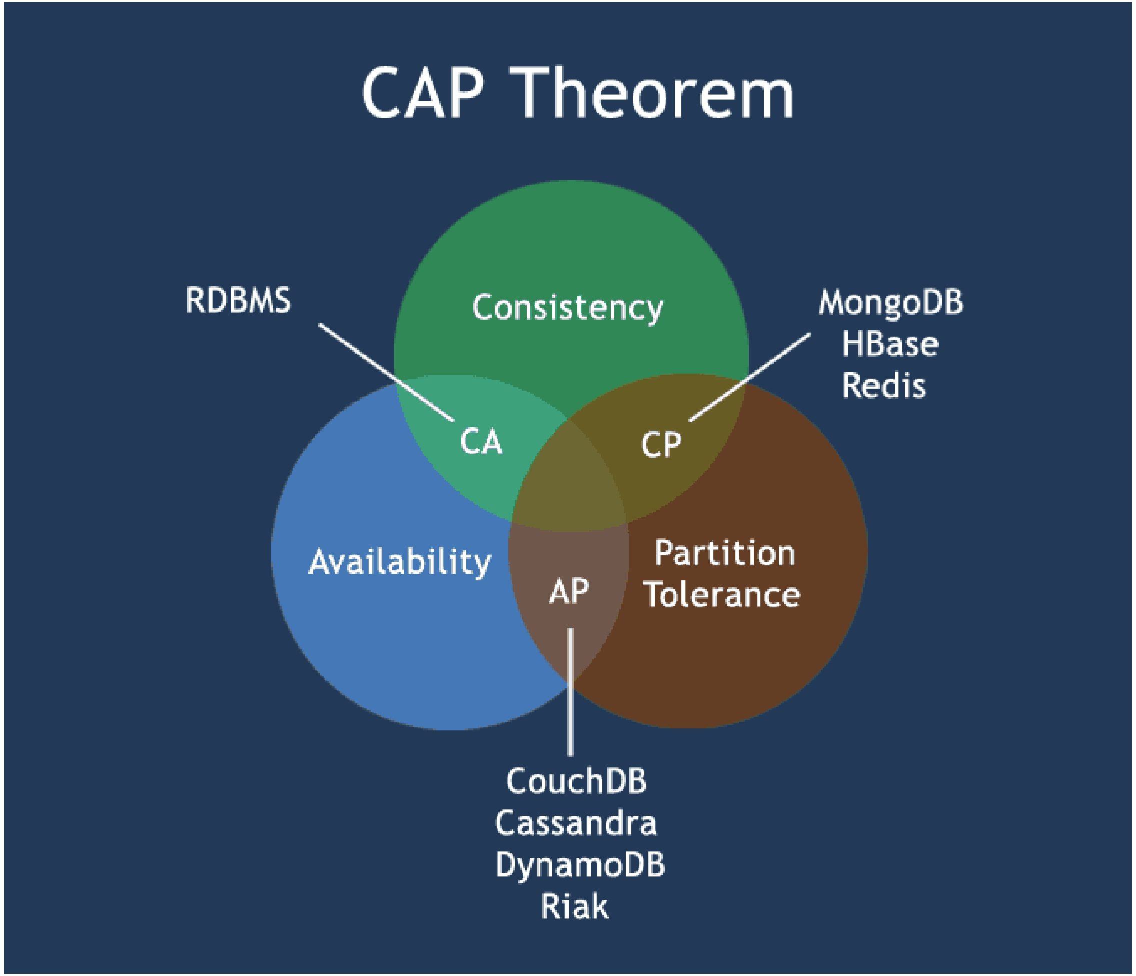
Algoritmo Cajero
Login
¿exitoso?
F
V
LoginError
Menú
menú
1
2
3
4
5
De Otro...
ConsultaSaldo
Depósito
Extracción
Transferencia
Salida
EntryError
FinAlgoritmo

TEOREMA CAP



TEOREMA CAP


Consistency
MongoDB
CA
CP
Availability
AP
Partition Tolerance
Cassandra


CAP Theorem
RDBMS
Consistency
MongoDB
HBase
Redis
CA
CP
Availability
AP
Partition
Tolerance
CouchDB
Cassandra
DynamoDB
Riak

TEOREMA CAP

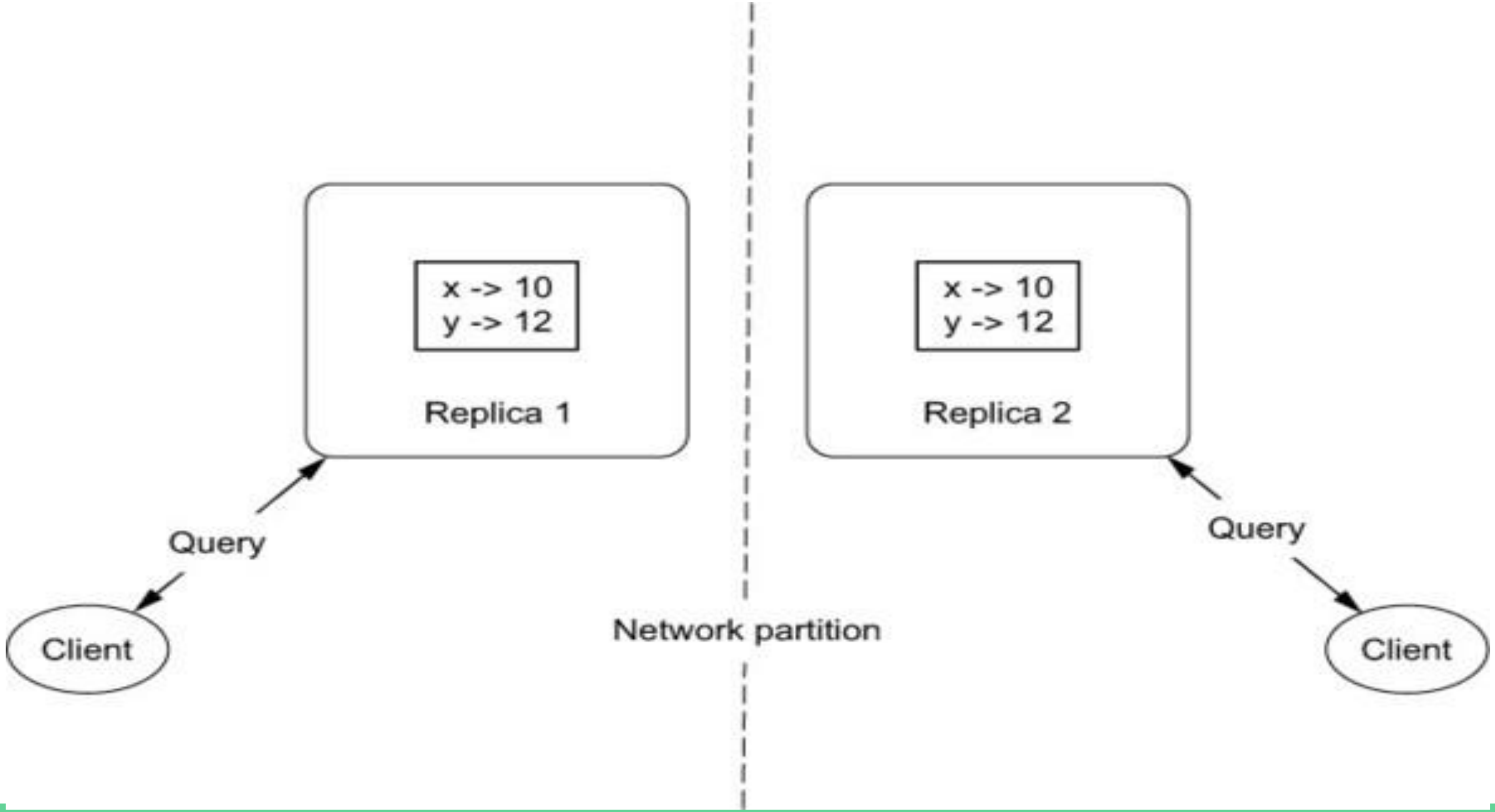
¿ Para la construcción de las siguientes bases de datos que parte del teorema CAP es importante ‘

- REDES SOCIALES
- SISTEMA BANCARIO

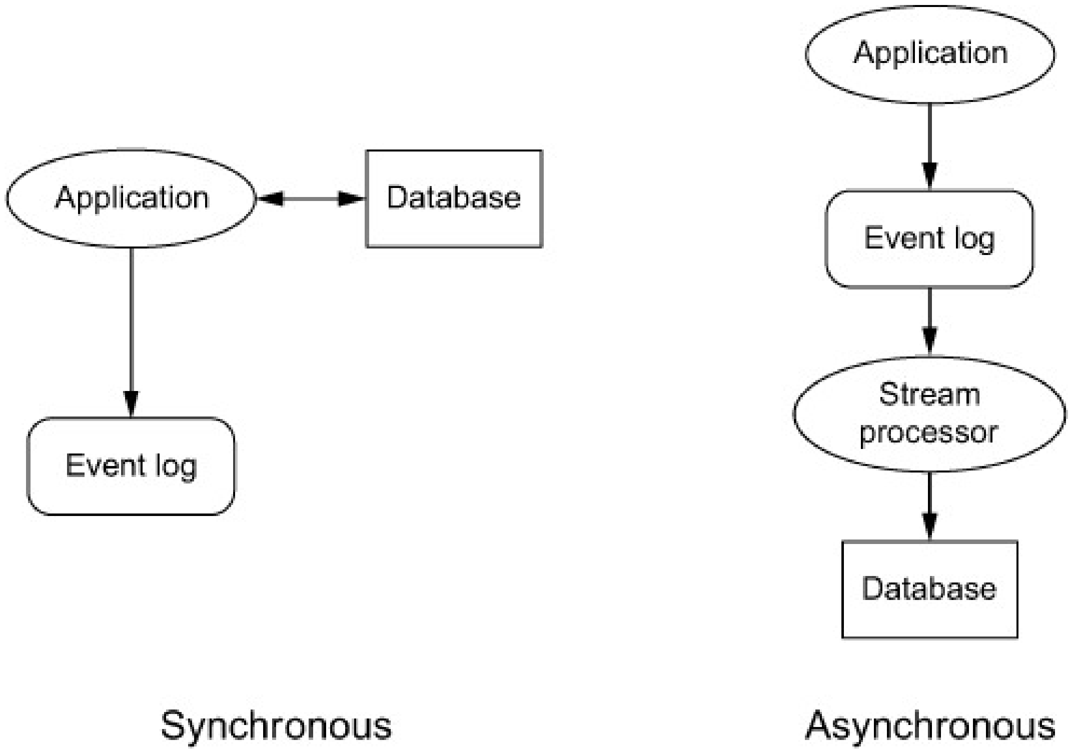
TEOREMA CAP y Google Spanner

	Google Cloud Spanner	Bases de datos relacionales	Bases de datos no relacionales
Esquema	+	+	-
SQL	+	+	-
Consistencia	fuerte	fuerte	final
Accesibilidad	alto	Tolerancia a fallos	alto
Escalabilidad	horizontal	vertical	horizontal
Replicación	automático	configurado	configurado

TEOREMA CAP


x -> 10
y -> 12
Replica 1
x -> 10
y -> 12
Replica 2
Query
Query
Client
Network partition
Client

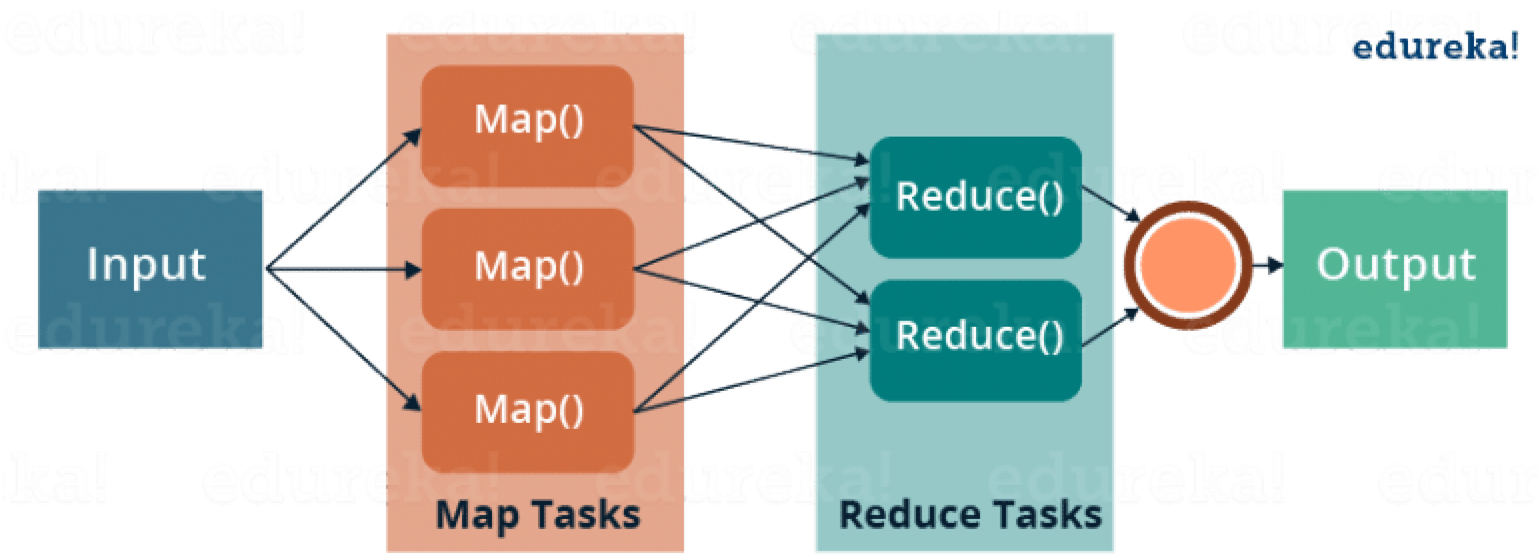
Falta de tolerancia a errores humanos



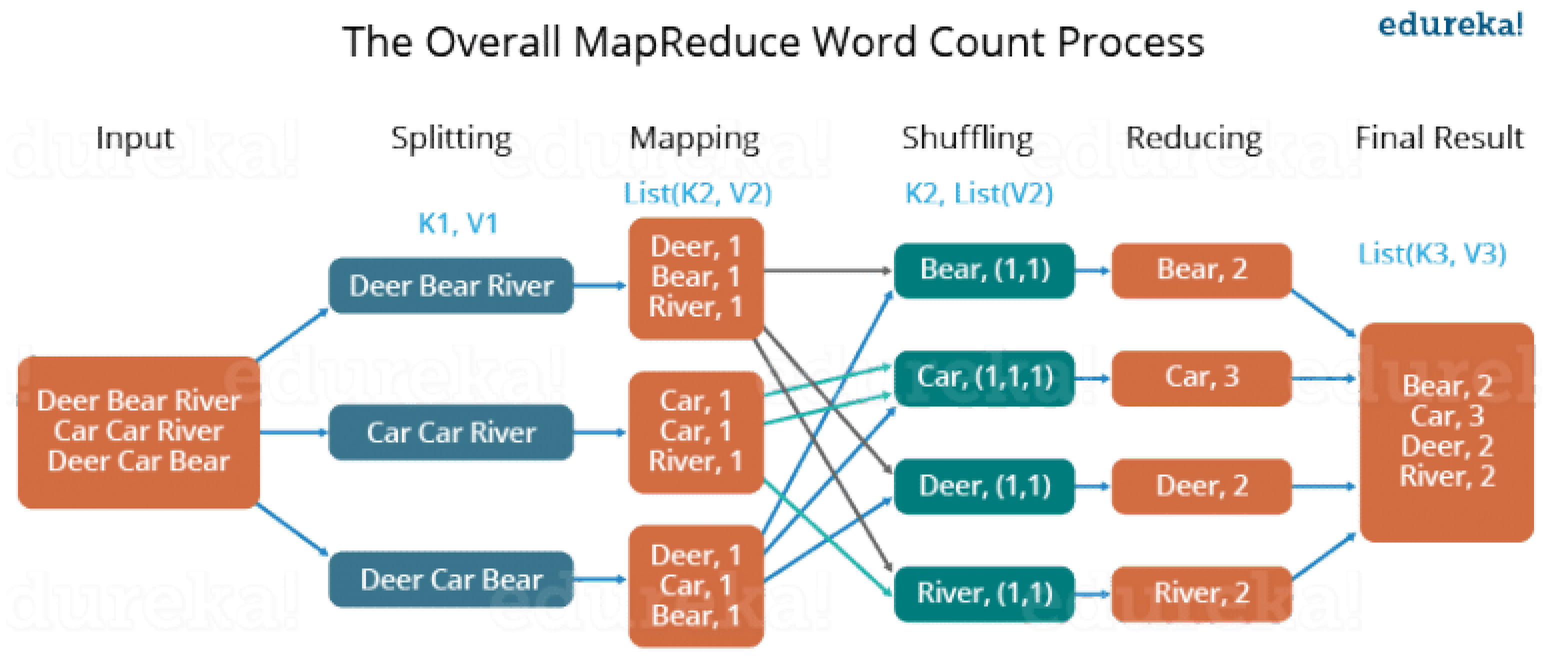
TENDENCIAS EN LA TECNOLOGÍA

- MAP REDUCE

MAP REDUCE


edureka!
Input
Map()
Map()
Map()
Map Tasks
Reduce()
Reduce()
Reduce Tasks
Output

MAP REDUCE



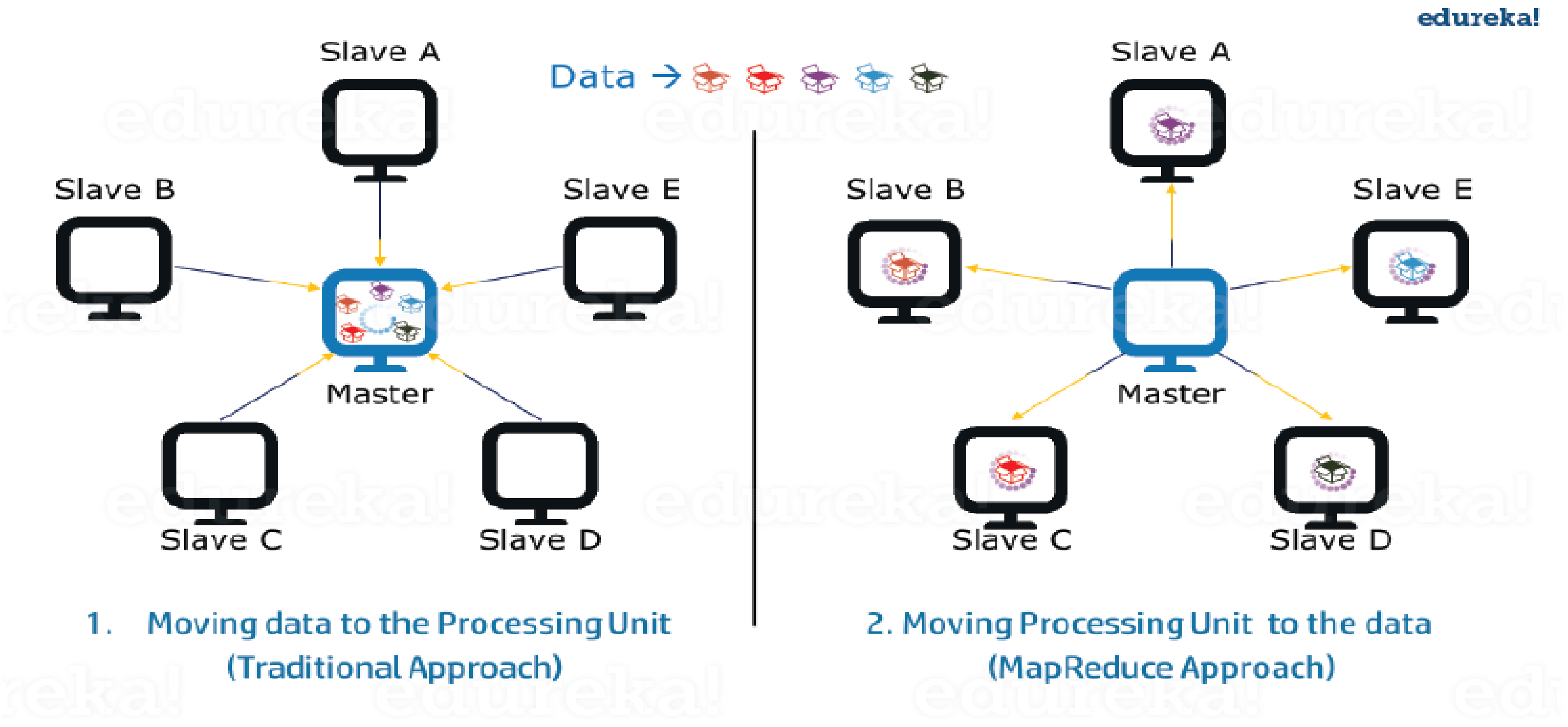
The Overall MapReduce Word Count Process
edureka!
Input
Splitting
Mapping
Shuffling
Reducing
Final Result
K1, V1
List(K2, V2)
K2, List(V2)
List(K3, V3)
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, (1,1)
Car, (1,1,1)
Deer, (1,1)
River, (1,1)
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

MAP REDUCE

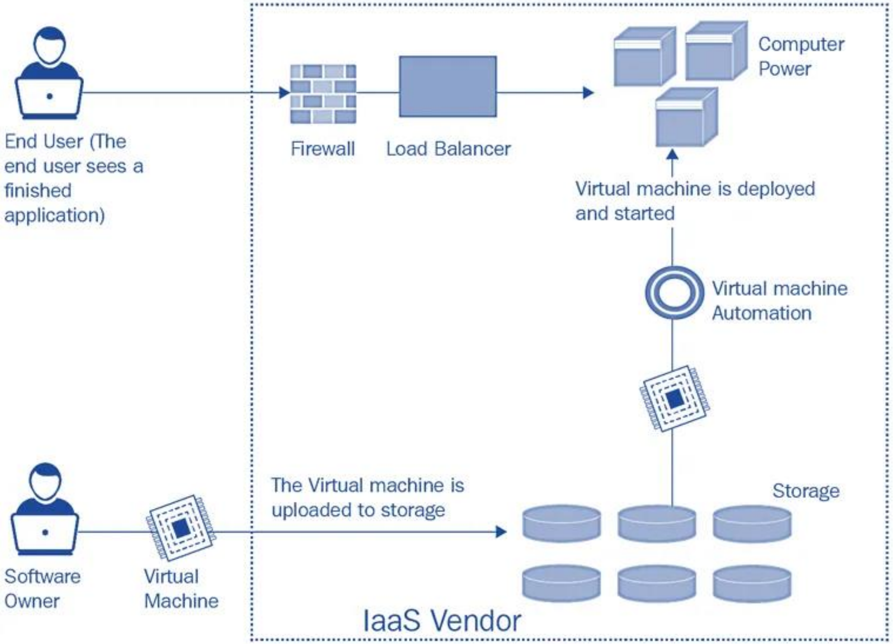
Input Text File

Key	**Value**
0	Dear Bear River
121	Car Car River
226	Deer Car Bear

MAP REDUCE: PROCESAMIENTO PARALELO

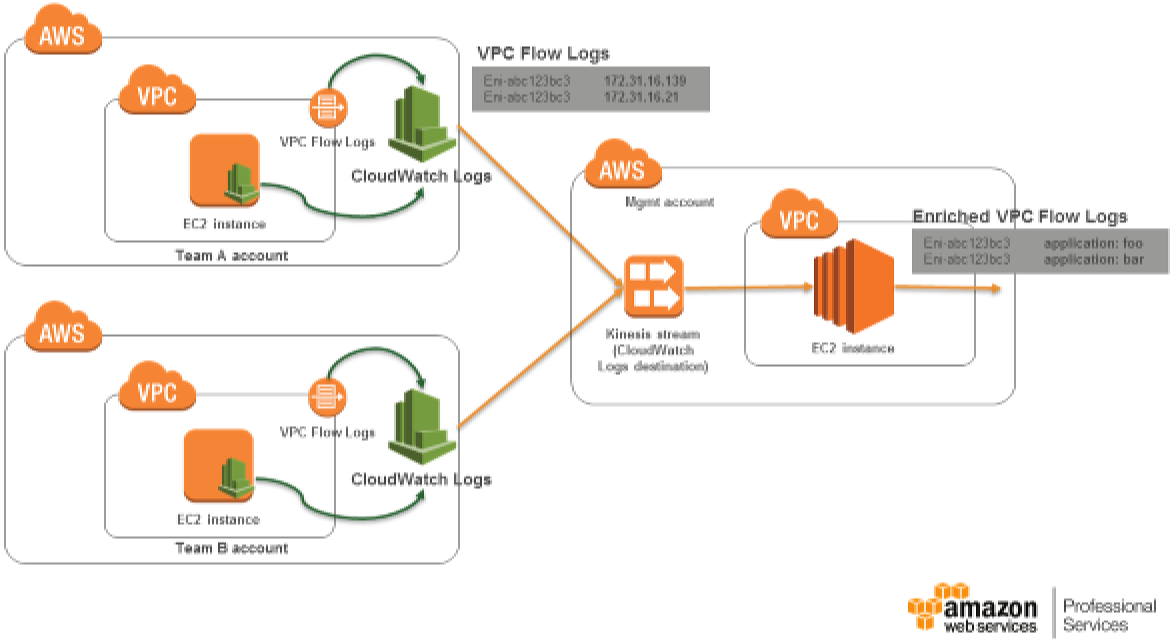


IAAS

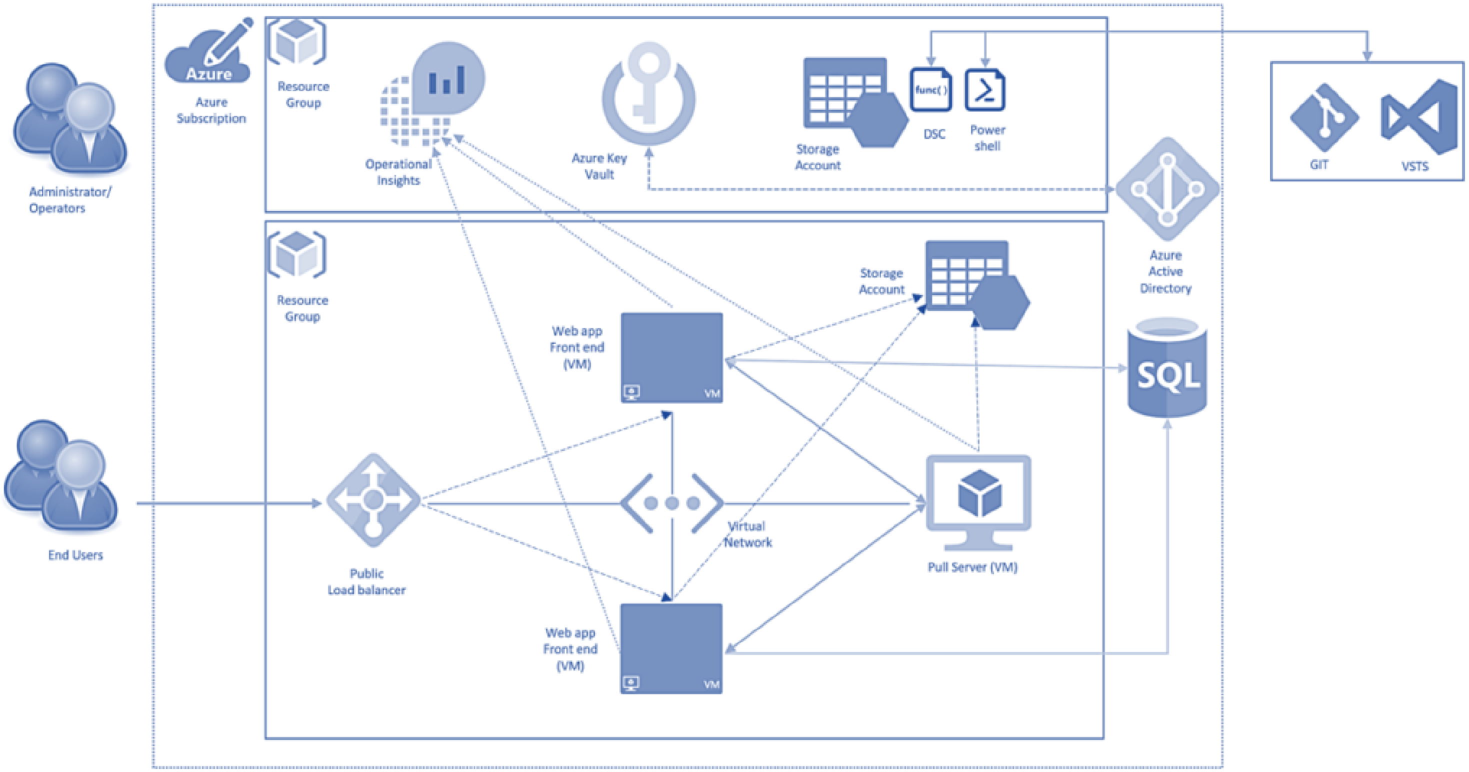


End User (The end user sees a finished application)
Firewall
Load Balancer
Computer Power
Virtual machine is deployed and started
Virtual machine Automation
Software Owner
Virtual Machine
The Virtual machine is uploaded to storage
Storage
IaaS Vendor

IAAS


AWS
VPC
VPC Flow Logs
Eni-abc123bc3 172.31.16.139
Eni-abc123bc3 172.31.16.21
VPC Flow Logs
CloudWatch Logs
EC2 instance
Team A account
AWS
VPC
VPC Flow Logs
CloudWatch Logs
EC2 Instance
Team B account
AWS
Mgmt account
VPC
Enriched VPC Flow Logs
Eni-abc123bc3 application: foo
Eni-abc123bc3 application: bar
Kinesis stream (CloudWatch Logs destination)
EC2 instance
amazon web services
Professional Services

IAAS

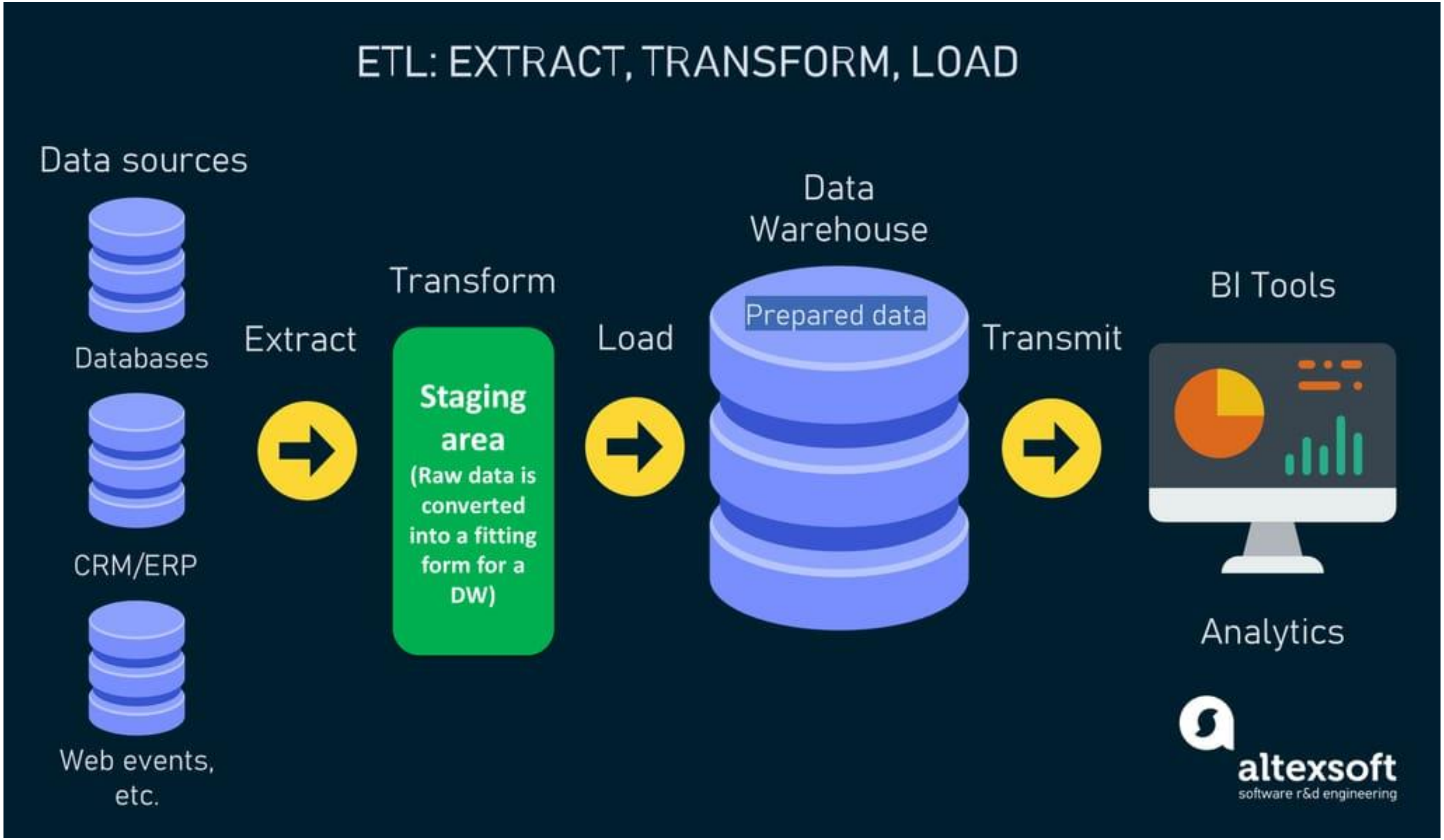


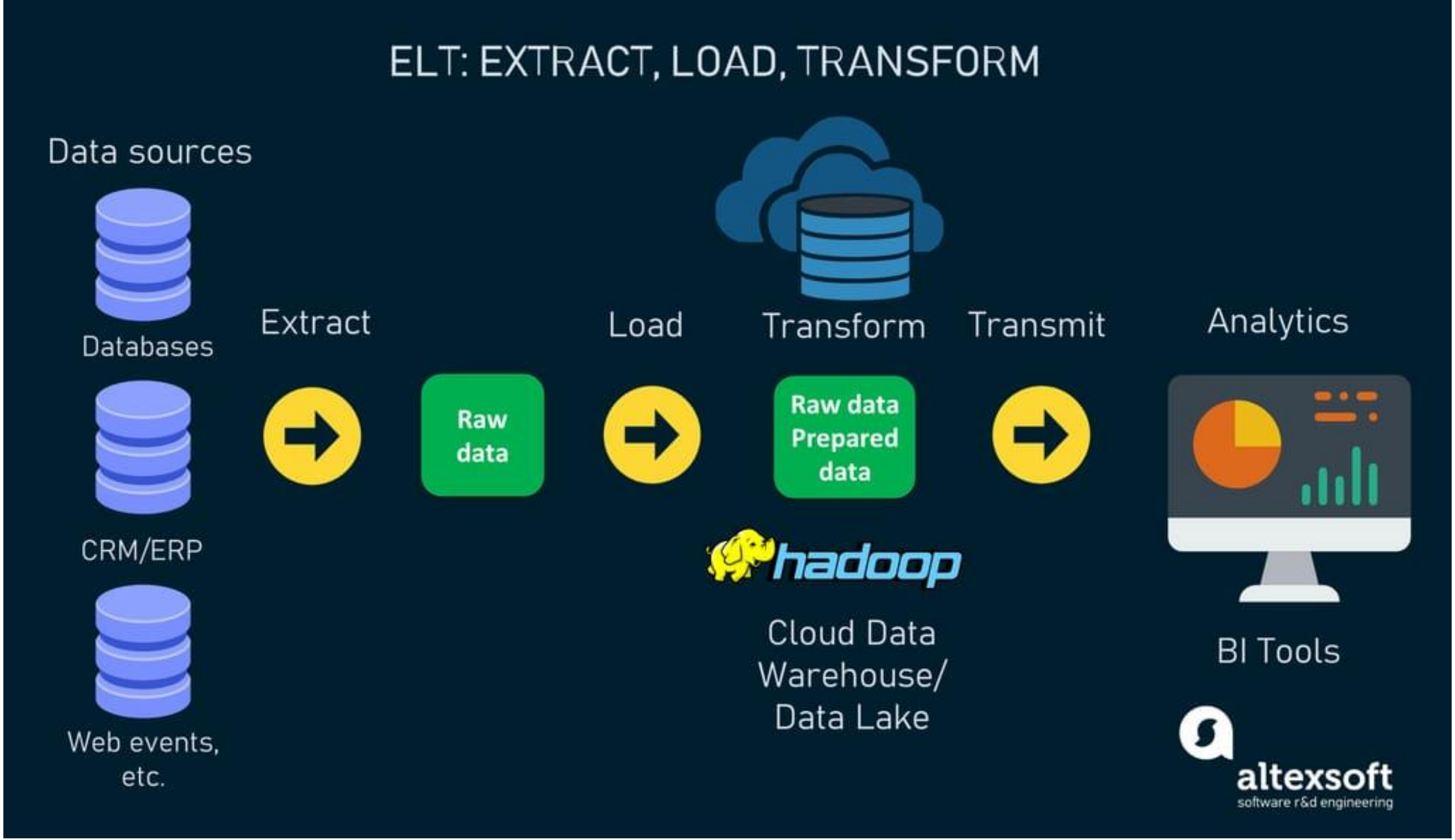
Azure
Azure Subscription
Resource Group
Operational Insights
Azure Key Vault
Storage Account
func()
DSC
Power shell
GIT
VSTS
Administrator/ Operators
Azure Active Directory
Resource Group
Storage Account
Web app Front end (VM)
VM
SQL
End Users
Public Load balancer
Virtual Network
Pull Server (VM)
Web app Front end (VM)
VM

IaaS

NUEVAS FUENTES DE DATOS

- **IoT (Internet of Things):** Sensores y dispositivos conectados que generan flujos de datos constantes.
- **Redes Sociales y Web:** Fuentes de datos no estructurados como tweets, publicaciones, reseñas en línea, etc.
- **Blockchain:** Registro distribuido de transacciones que se está utilizando cada vez más como fuente de datos en sectores como las finanzas y la cadena de suministro.


ETL: EXTRACT, TRANSFORM, LOAD
Data sources
Databases
CRM/ERP
Web events, etc.
Extract
Transform
Staging area
(Raw data is converted into a fitting form for a DW)
Load
Data Warehouse
Prepared data
Transmit
BI Tools
Analytics
altexsoft
software r&d engineering

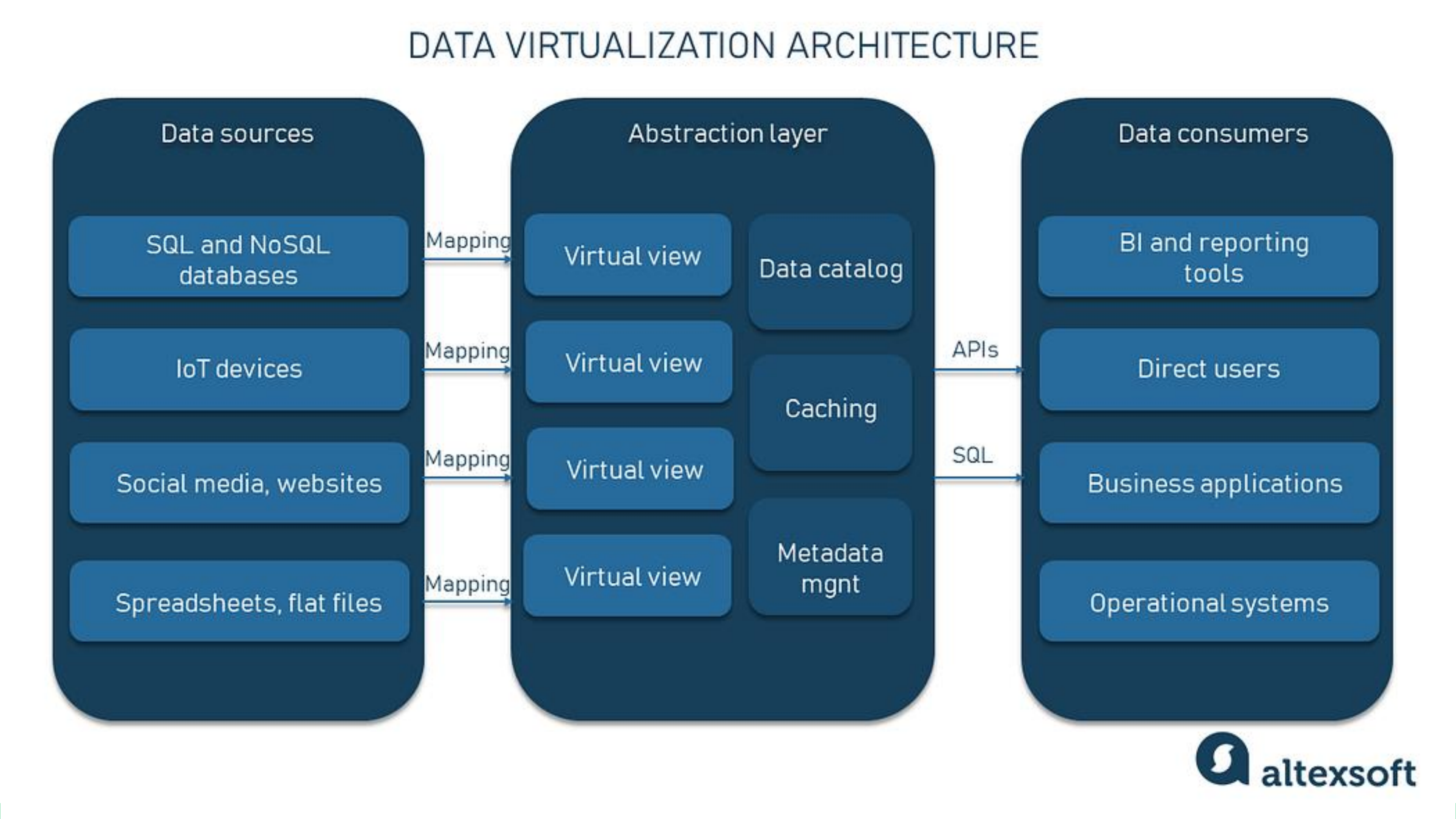


ELT: EXTRACT, LOAD, TRANSFORM
Data sources
Databases
CRM/ERP
Web events, etc.
Extract
Raw data
Load
Transform
Raw data
Prepared data
hadoop
Cloud Data Warehouse/ Data Lake
Transmit
Analytics
BI Tools
altexsoft
software r&d engineering

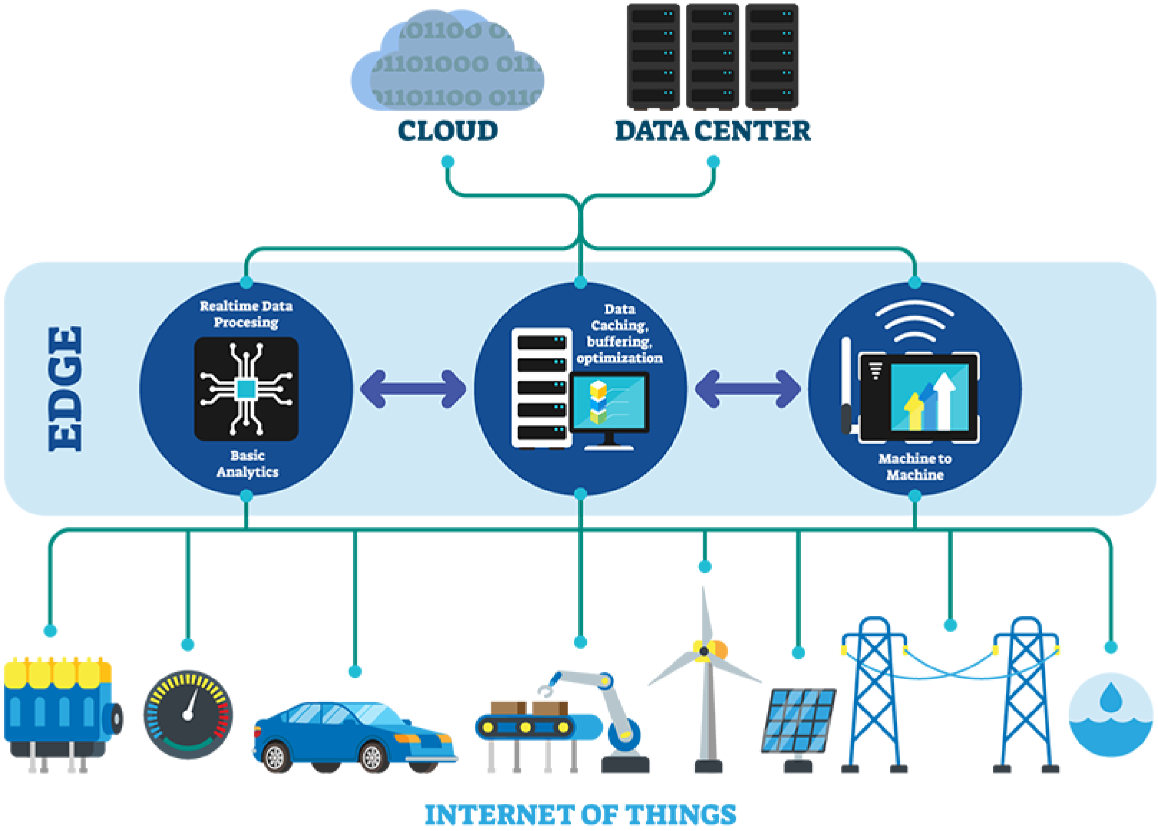
ETL VS ELT PROCESSES COMPARED

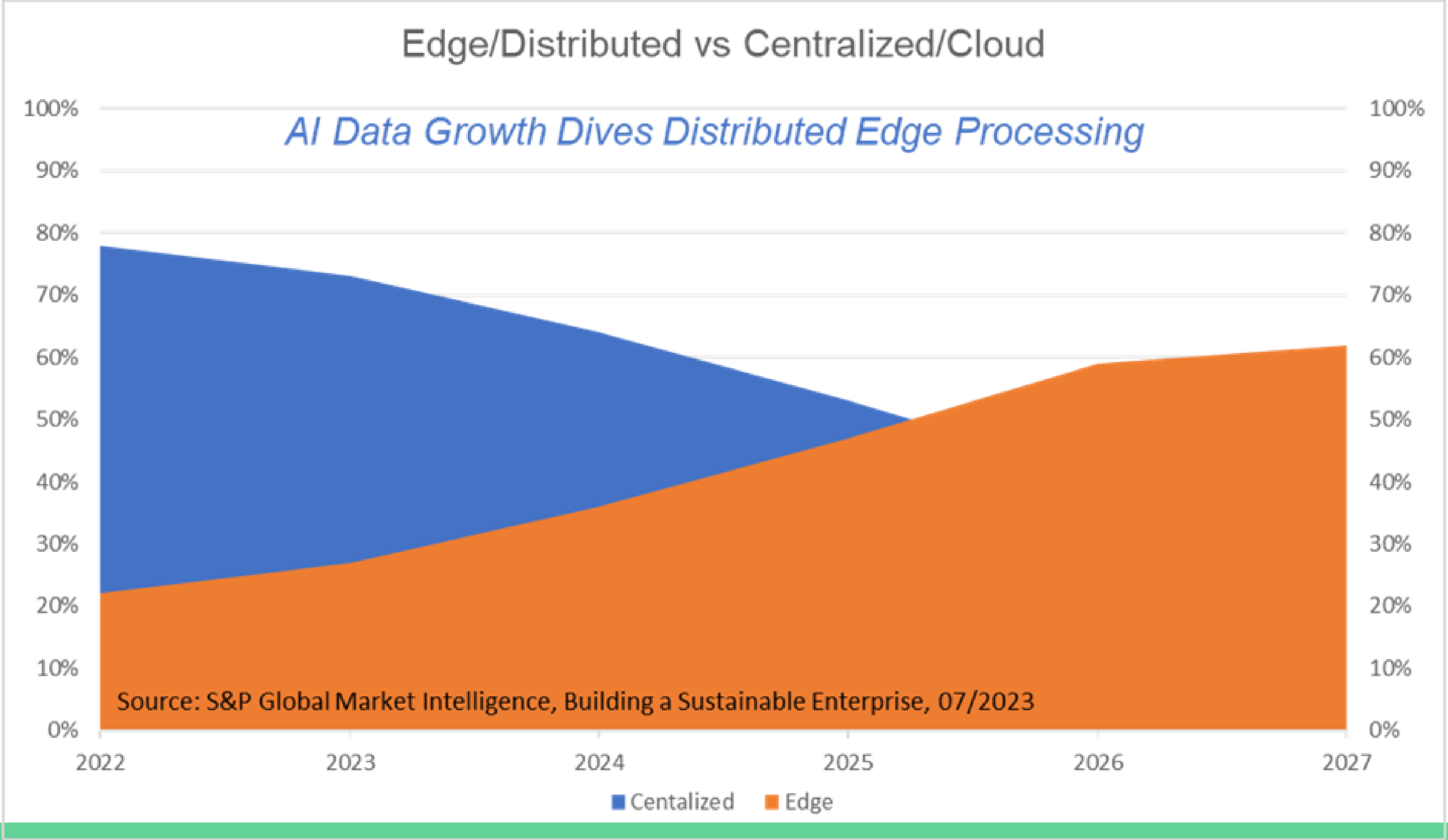
	ETL	ELT
Stands for	Extract, Transform, Load	Extract, Load, Transform
Deployment	Cloud-based / on-premises	Cloud-based
Technology maturity	Reliable and mature	Relatively new
Large volumes of data	ETL makes it more difficult to work with large volumes of data	ELT makes it easier to work with large volumes of data
Data type	Mostly structured data	All data: structured and unstructured
Target system	Onsite / cloud data warehouses	Cloud data warehouses / data lakes
Costs	Expensive (on-premises) Cost-efficient (cloud)	Cost-efficient
Maintenance	ETL requires higher maintenance when performed in a traditional way Cloud ETL isn't maintenance-intensive	ELT requires little maintenance
Load times	Data loading is slower	Data loading is faster
Transformation times	Significantly slower since transformations are performed on a separate server	Faster since transformations happen inside a target system on-demand
Compliance	Allows for sensitive data protection, encryption, redaction before it gets into a system	Requires uploading all data into a system without any redaction/removing sensitive data
Tools	Informatica, Cognos, Oracle, IBM	Kafka, Hevo Data, Talend
Expertise needed	Vast experience in performing data sourcing, exportation, transformation. ETL specialists, engineers, analysts	Deep knowledge of existing tools and strong niche skills. ELT specialists are more difficult to find
Better for	• Small portions of data • Structured data • Legacy systems, relational databases	• Getting all raw data in a system quickly • Unstructured data • Projects with a tendency to scale

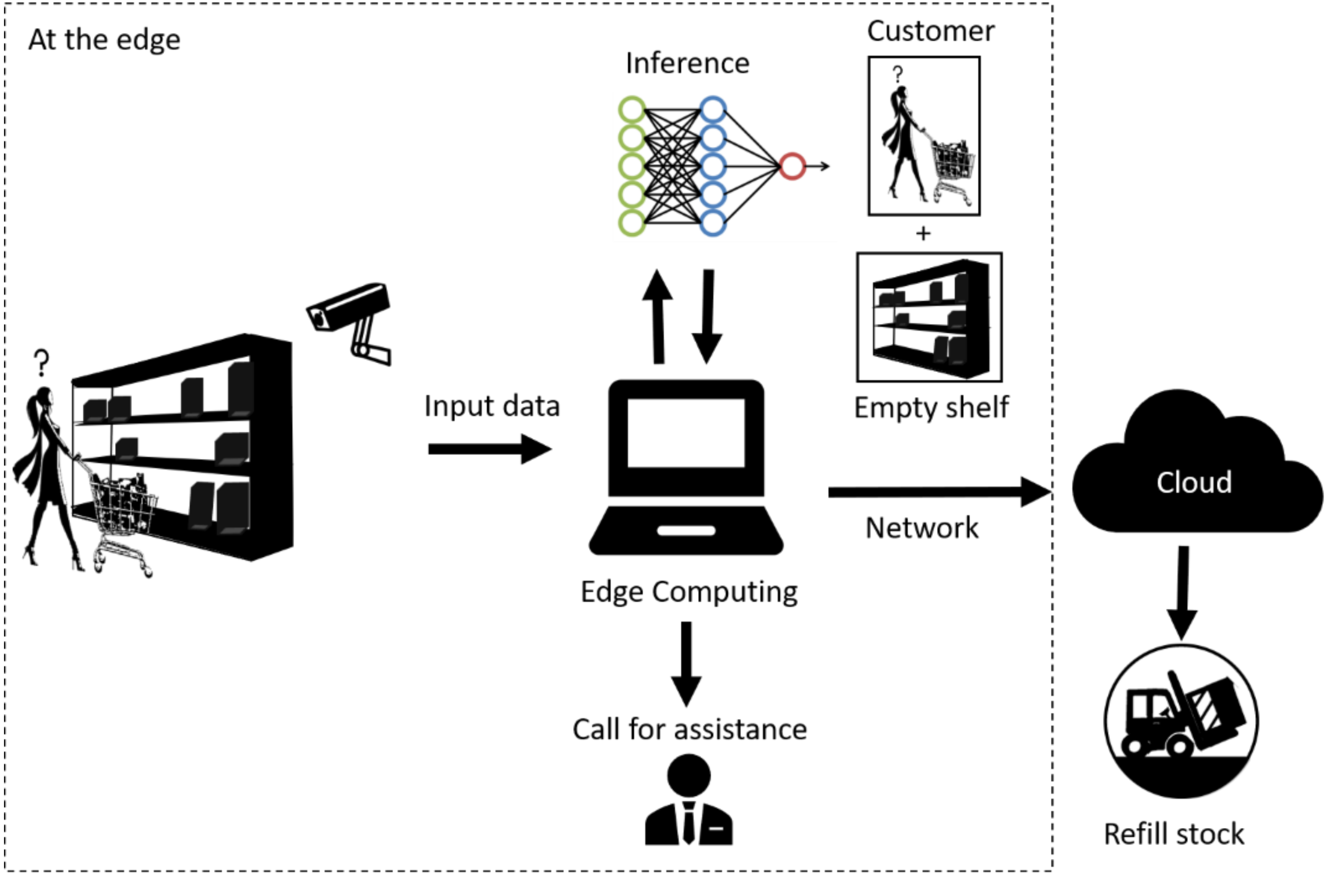
altexsoft
software r&d engineering

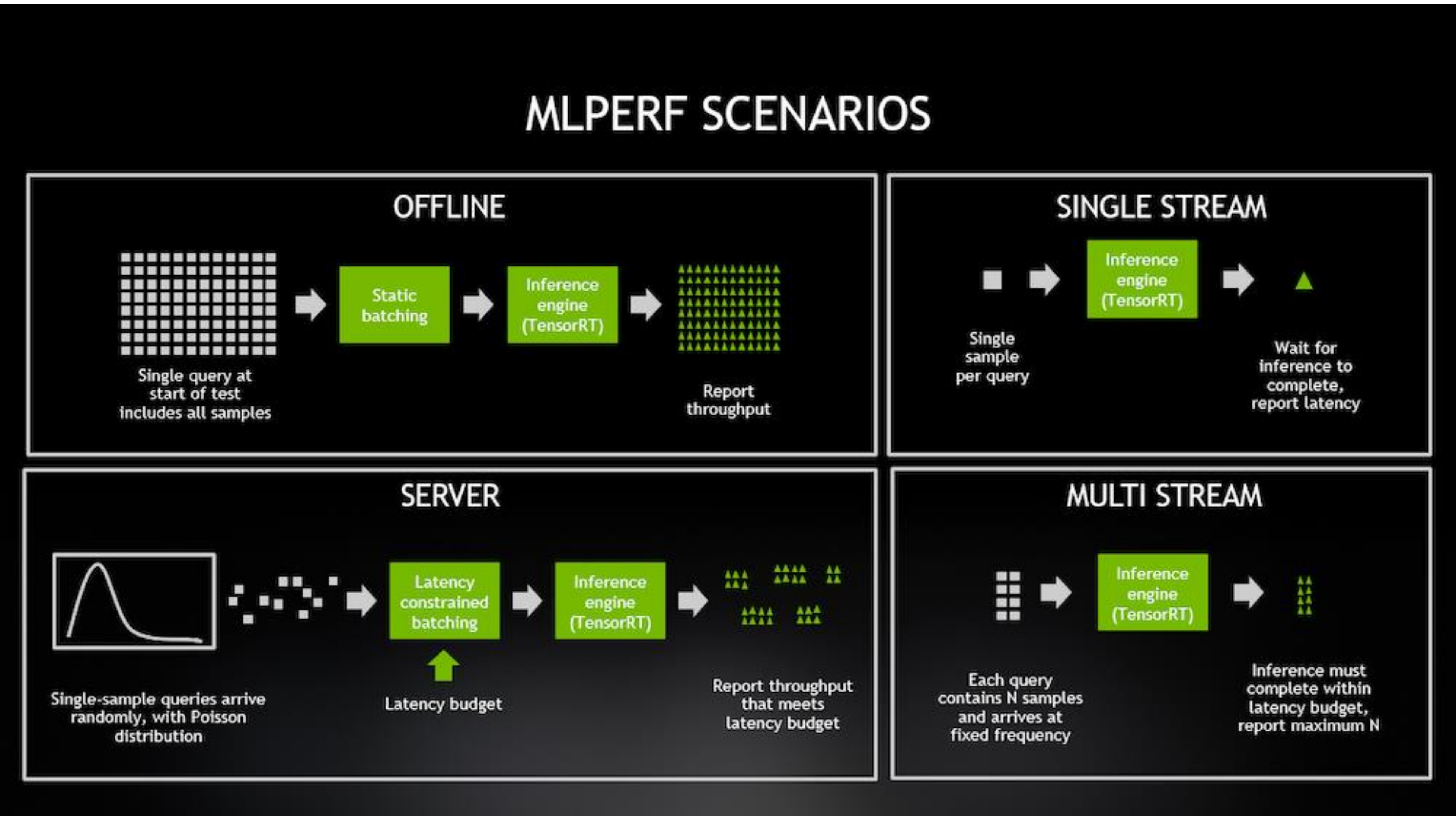

DATA VIRTUALIZATION ARCHITECTURE
Data sources
SQL and NoSQL databases
IoT devices
Social media, websites
Spreadsheets, flat files
Mapping
Mapping
Mapping
Mapping
Abstraction layer
Virtual view
Virtual view
Virtual view
Virtual view
Data catalog
Caching
Metadata mgnt
APIs
SQL
Data consumers
BI and reporting tools
Direct users
Business applications
Operational systems
altexsoft

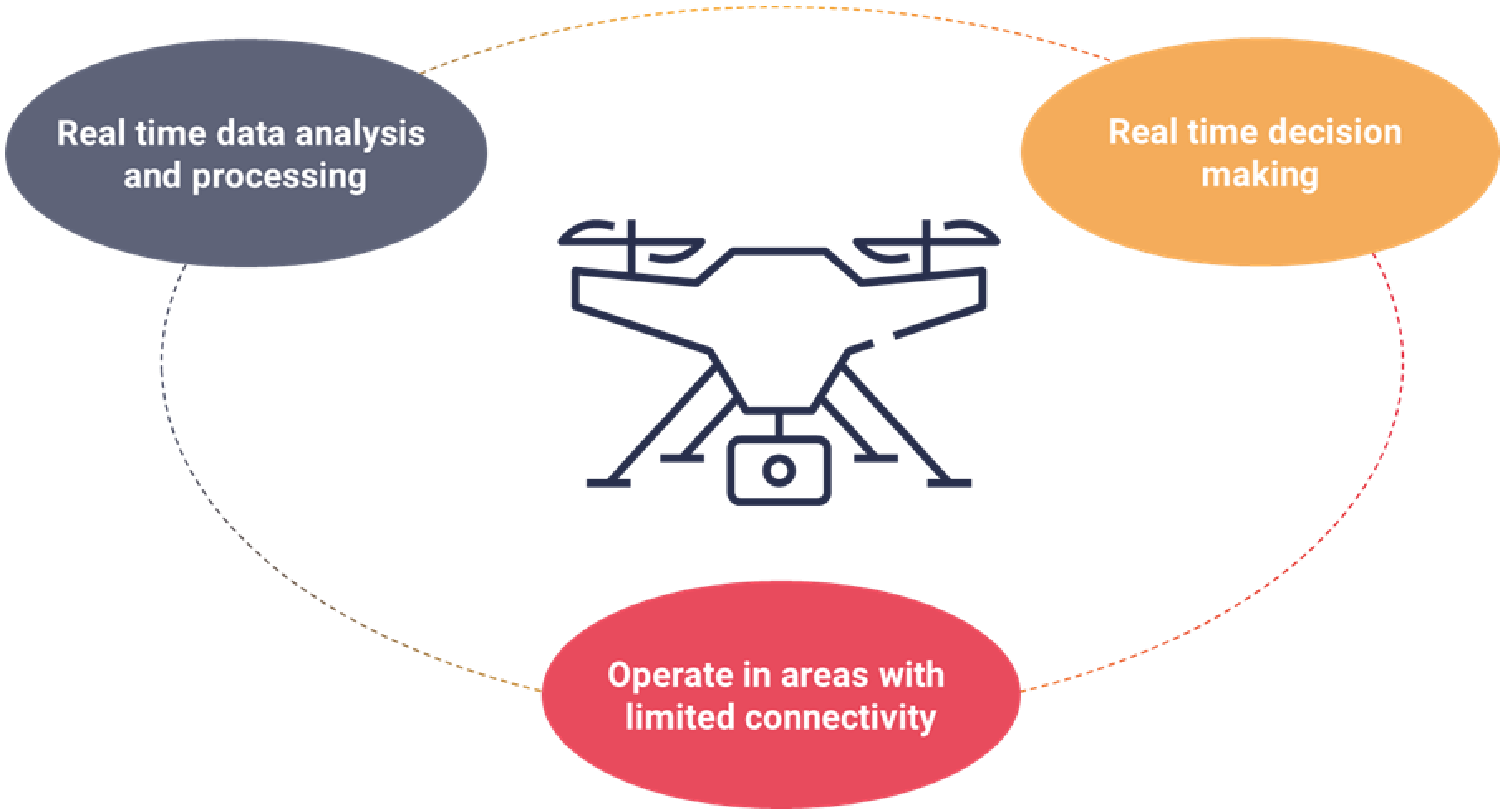
Edge Computing



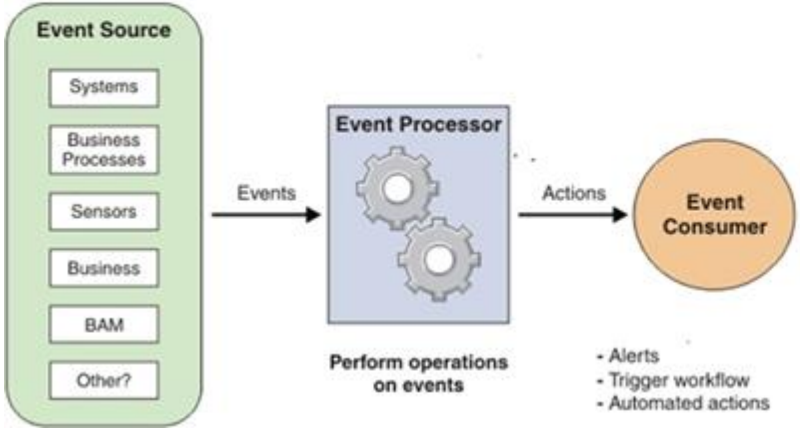

Edge/Distributed vs Centralized/Cloud
AI Data Growth Dives Distributed Edge Processing
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Source: S&P Global Market Intelligence, Building a Sustainable Enterprise, 07/2023
2022
2023
2024
2025
2026
2027
Centalized
Edge

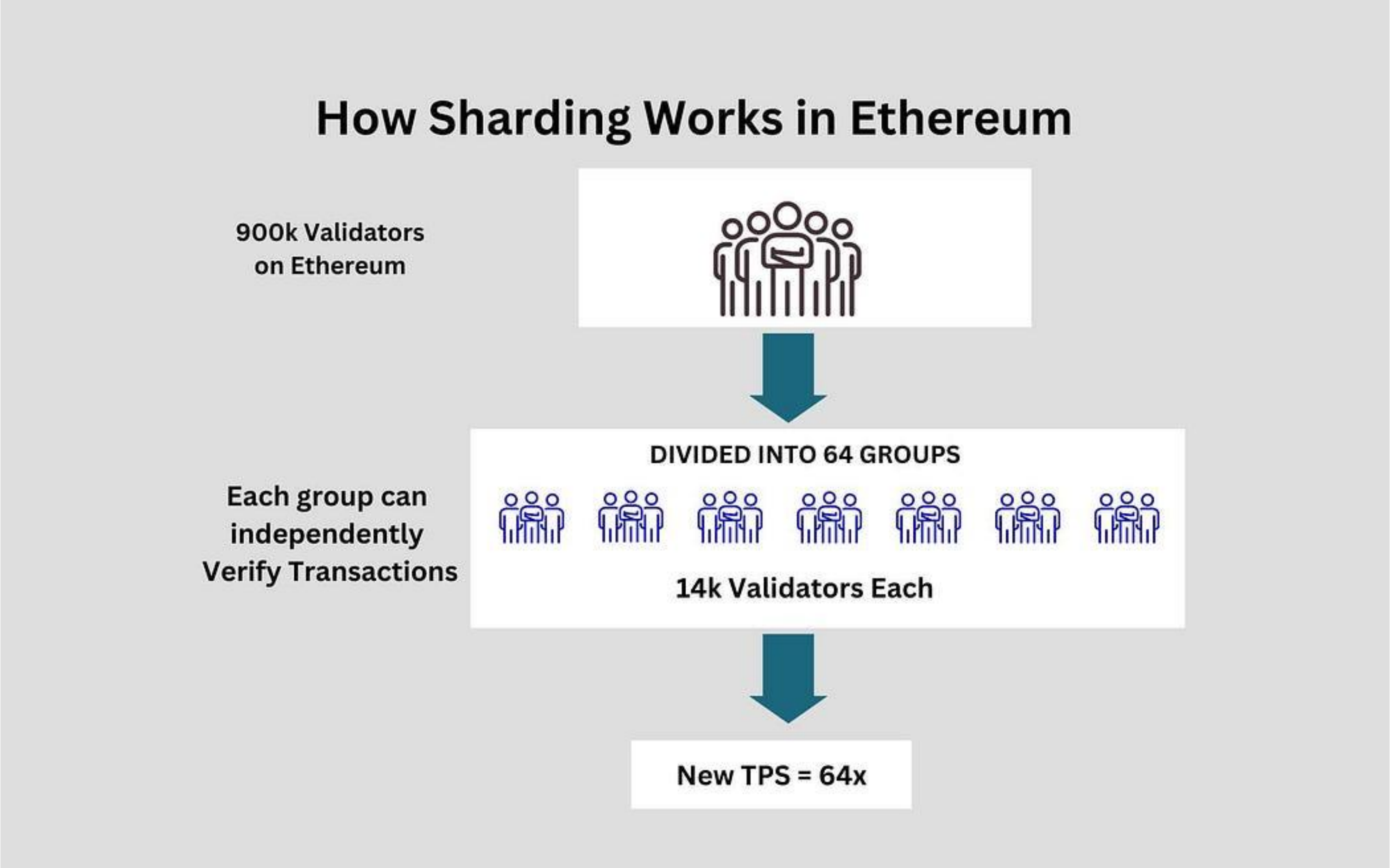

At the edge
Inference
Customer
?
+
Empty shelf
?
Input data
Edge Computing
Network
Cloud
Call for assistance
Refill stock

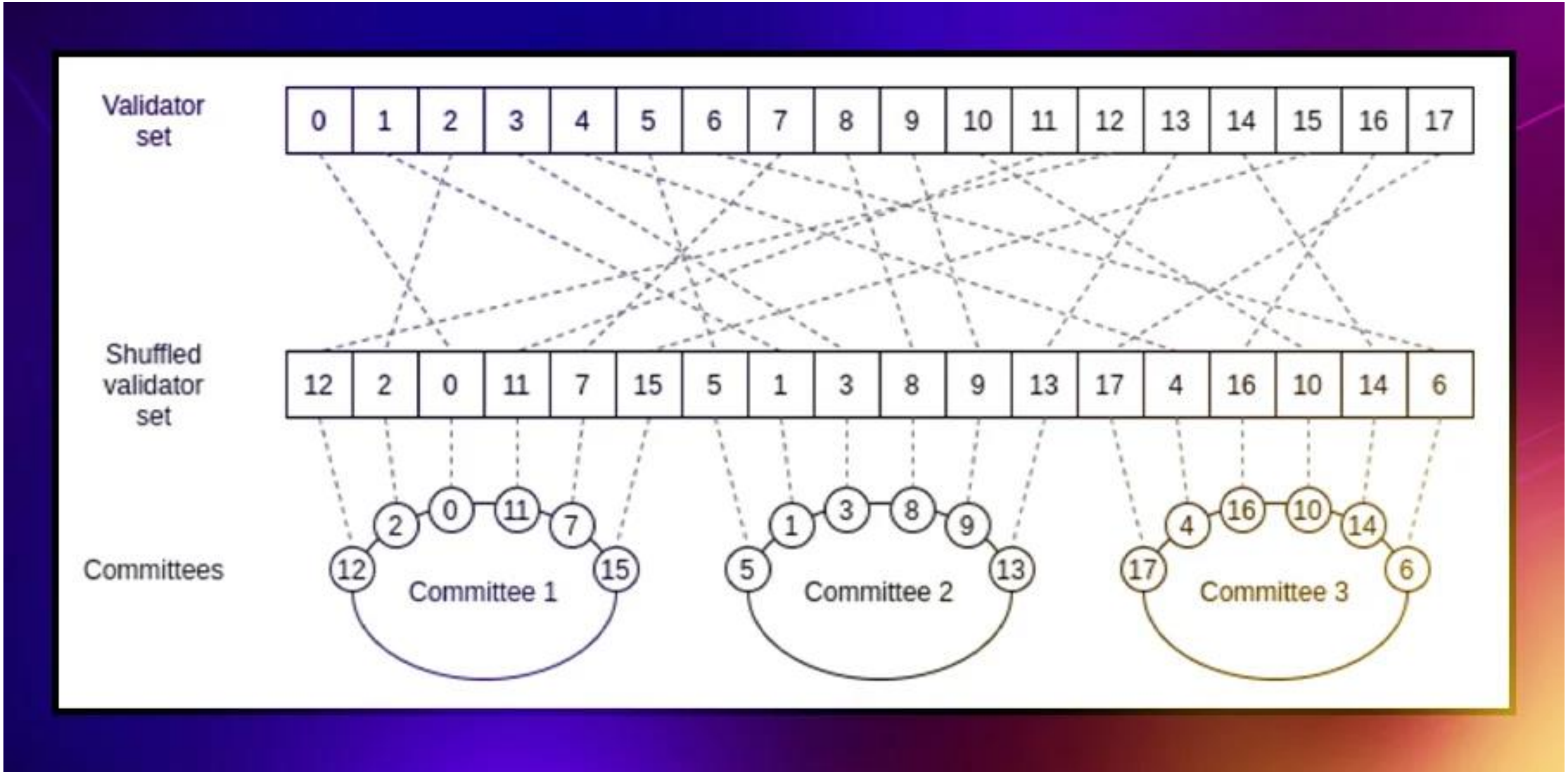

MLPERF SCENARIOS
OFFLINE
Static batching
Inference engine (TensorRT)
Single query at start of test includes all samples
Report throughput
SINGLE STREAM
Inference engine (TensorRT)
Single sample per query
Wait for inference to complete, report latency
SERVER
Latency constrained batching
Inference engine (TensorRT)
Latency budget
Single-sample queries arrive randomly, with Poisson distribution
Report throughput that meets latency budget
MULTI STREAM
Inference engine (TensorRT)
Each query contains N samples and arrives at fixed frequency
Inference must complete within latency budget, report maximum N


Real time data analysis and processing
Real time decision making
Operate in areas with limited connectivity

Complex Event Processing [CEP]




How Sharding Works in Ethereum
900k Validators on Ethereum
DIVIDED INTO 64 GROUPS
Each group can independently Verify Transactions
14k Validators Each
New TPS = 64x



Validator set
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
Shuffled validator set
12 2 0 11 7 15 5 1 3 8 9 13 17 4 16 10 14 6
Committees
12 2 0 11 7 15
Committee 1
5 1 3 8 9 13
Committee 2
17 4 16 10 14 6
Committee 3

$$\frac{365}{365}\times\frac{364}{365}\times\frac{363}{365}\times\cdots\times\frac{344}{365}\times\frac{343}{365}=49.27\%$$

1 2 3 4 – 21 22 23

Preguntas y discusión

- **Espacio para preguntas:** Tiempo dedicado a resolver dudas y preguntas de los estudiantes sobre los temas tratados en la clase.

- **Discusión abierta:** Fomentar una discusión abierta sobre el futuro del DataWarehouse y su impacto en diferentes industrias.

Photo by SOULSANA on Unsplash