
plain concepts

DÍA 5: AGENDA

- Kylin
- Kafka
- PolyBase
- Azure Data Lake
- Azure Data Factory
- **Ejercicio práctico del día**

KYLIN

plain concepts

OLAP SOBRE HADOOP: KYLIN

- Construcción de cubos
- Lenguaje de consultas
- Conectividad con herramientas de cliente

¿QUE ES KYLIN?

- Kylin = Motor de análisis distribuido open source
- Proporciona:
  - Capacidad de análisis en grandes datasets
  - ANSI SQL query interface
  - Análisis multidimensional
- Proyecto de primer nivel de Apache
- Originario de eBay

OBJETIVOS INICIALES

- Baja latencia (<1sec) con billones de registros
- Uso de estándar ANSI SQL
- Ofrecer funcionalidades OLAP avanzadas
- Integración con herramientas BI
- Soporte para alta cardinalidad en dimensiones
- Alta concurrencia
- Arquitectura distribuida para proceso de grandes volúmenes de datos

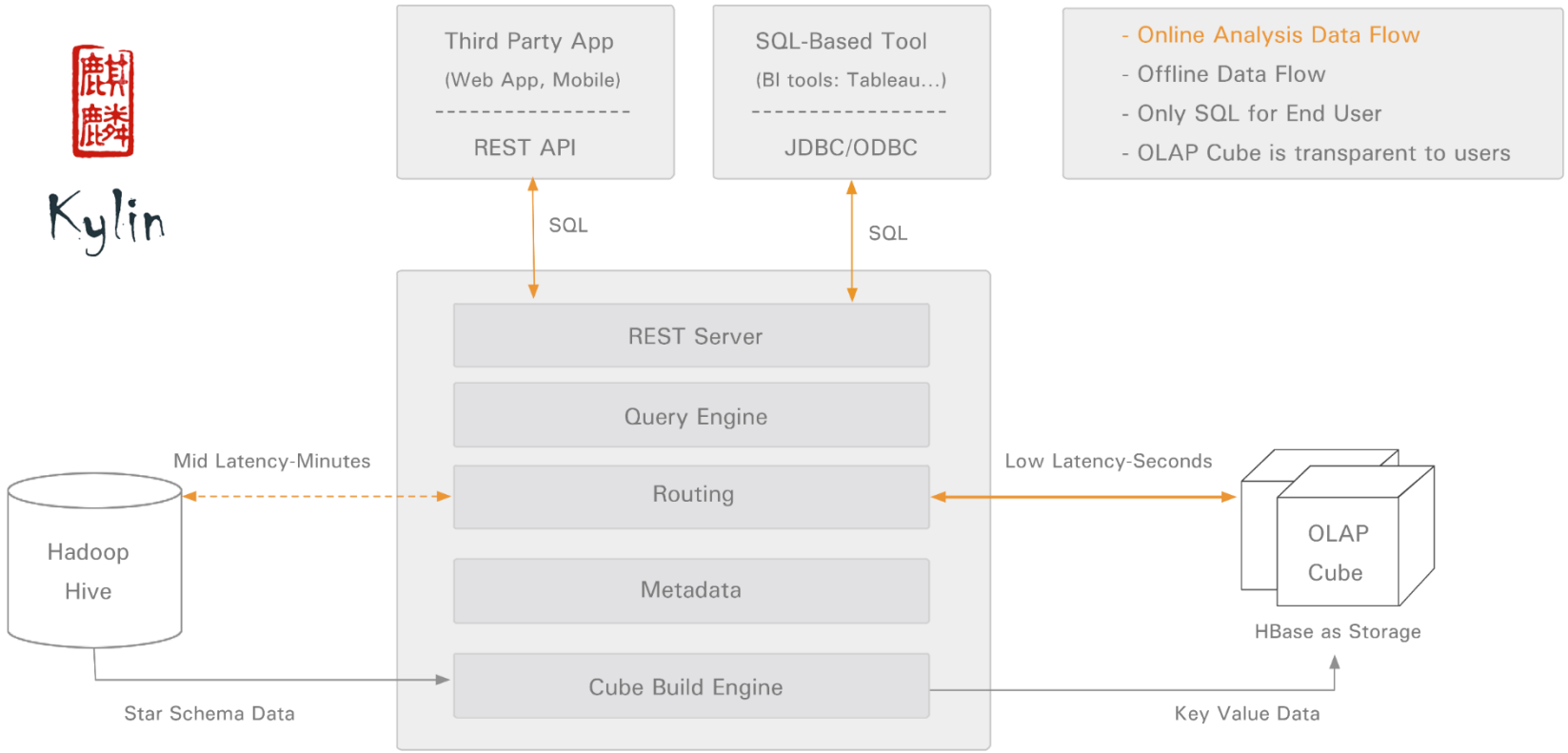
ESTRATEGIA

- Usar Calcite como interprete SQL
  - SQL
  - Optimizador CBO (Cost-Based Optimizer)
  - Enlaza Kylin con Apache Drill y con versiones futuras de Hive
- Construir los cubos independientes del origen, ahora Hive y en un futuro Spark
- Las agregaciones generadas se almacenan en una única maquina

RETOS TECNICOS

- Gran volumen de datos
- Joins de grandes tablas
- Análisis de diferentes niveles de agregación
- Map Reduce jobs

ARQUITECTURA



ECOSISTEMA

- Hive
  - Fuente de datos, pre join star schema en proceso del cubo
- MapReduce
  - Agrega las medidas en el proceso del cubo
- HDFS
  - Almacena los archivos temporales en el proceso
- HBase
  - Almacena los cubos y es la Fuente de datos de las consultas
- Calcite
  - SQL parsing, generación y optimización de código

CARACTERISTICAS DESTACADAS

ANSI SQL Interface	Integración con herramientas de BI	Acceso a datos en Hadoop con latencias < 1 segundo	MOLAP Cube
Soporta origenes >10 billones de filas	Soporta compresión	Proceso incremental de cubos	Gestión y monitorización de jobs de carga
Interfaz web	Seguridad a nivel de Proyecto y cubo	Soporta LDAP	REST API

plain concepts

KYLIN

APACHE KAFKA

plain concepts

¿QUÉ ES APACHE KAFKA?

- Kafka es un sistema de mensajes basado en el mecanismo “publish-subscribe”
- Desarrollado originalmente por LinkedIn
- Implementado en Scala/Java
- Proyecto Apache de primer nivel desde 2012
- Usado en muchas empresas: LinkedIn, Netflix, Twitter, Spotify, Loggly, Mozilla, Airbnb, Cisco, Gnip, InfoChimps, Ooyala, Square, Uber

¿QUÉ ES APACHE KAFKA?

- La motivación de LinkedIn para crear Kafka fue
  - “Una plataforma unificada para gestionar todas las fuentes de datos en tiempo real de las que dispone una gran empresa”
- Características
  - Alto rendimiento para soportar fuentes de eventos de gran volumen
  - Soporte para procesar estas fuentes de eventos en tiempo real para crear fuentes de datos derivadas
  - Soporte de carga de datos atrasados para gestionar ingestas periódicas desde sistemas offline
  - Tolerancia a fallos garantizada en caso de errores hardware

PILARES

Rápido

Escalable

Durable

Distribuido

PILARES

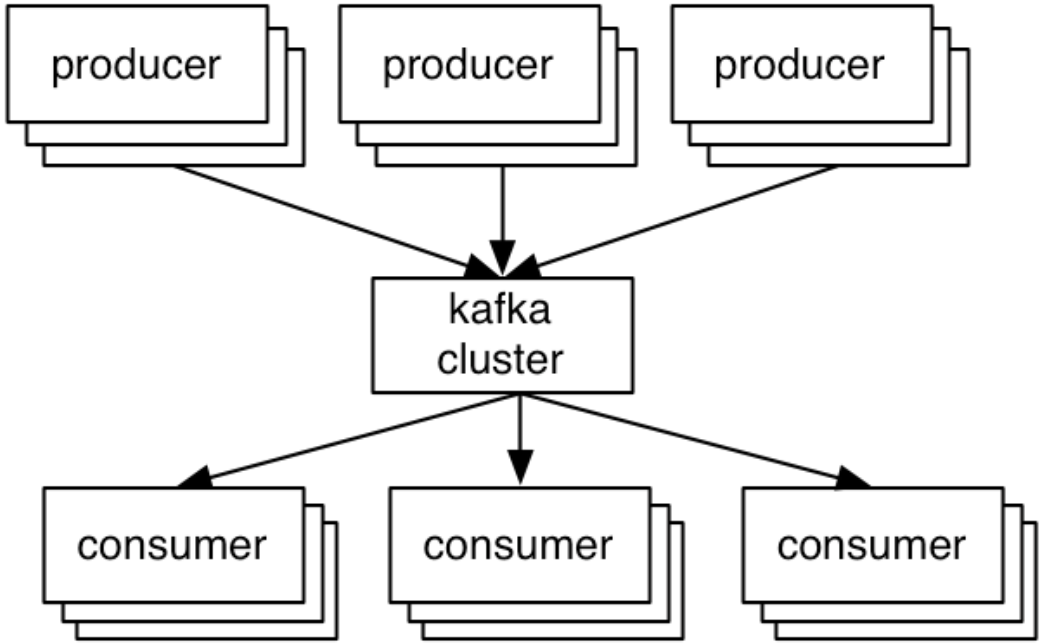
- Rápido
  - Un solo broker Kafka puede gestionar cientos de MB por segundo como tasa de lectura y escritura, desde miles de clientes
- Escrituras rápidas
  - Pese a que Kafka persiste todos los datos a disco, las escrituras van directamente a la cache de página del Sistema operativo, es decir, la memoria RAM
- Lecturas rápidas
  - Gran eficiencia transfiriendo datos desde la cache de página a un socket de red

PILARES

- Escalable
  - Diseñado para que un solo cluster pueda servir como la columna vertebral de una gran empresa a nivel de datos. Posibilidad de escalarlo y expandirlo de forma transparente y sin downtime
- Durable
  - Los mensajes se persisten en disco y se replican dentro del cluster para prevenir perdidas de datos. Cada broker maneja Tb de mensajes sin impacto en el rendimiento
- Distribuido
  - Diseño basado en clusters que ofrece garantías de Resistencia y tolerancia a fallos

ARQUITECTURA BÁSICA

- Los actores
  - Los Productores escriben en los Brokers
  - Los Consumidores leen de los Brokers
  - Todo en una arquitectura distribuida
- Los datos
  - Los datos se almacenan en Topics
  - Los Topics se separan en Particiones, que están replicadas

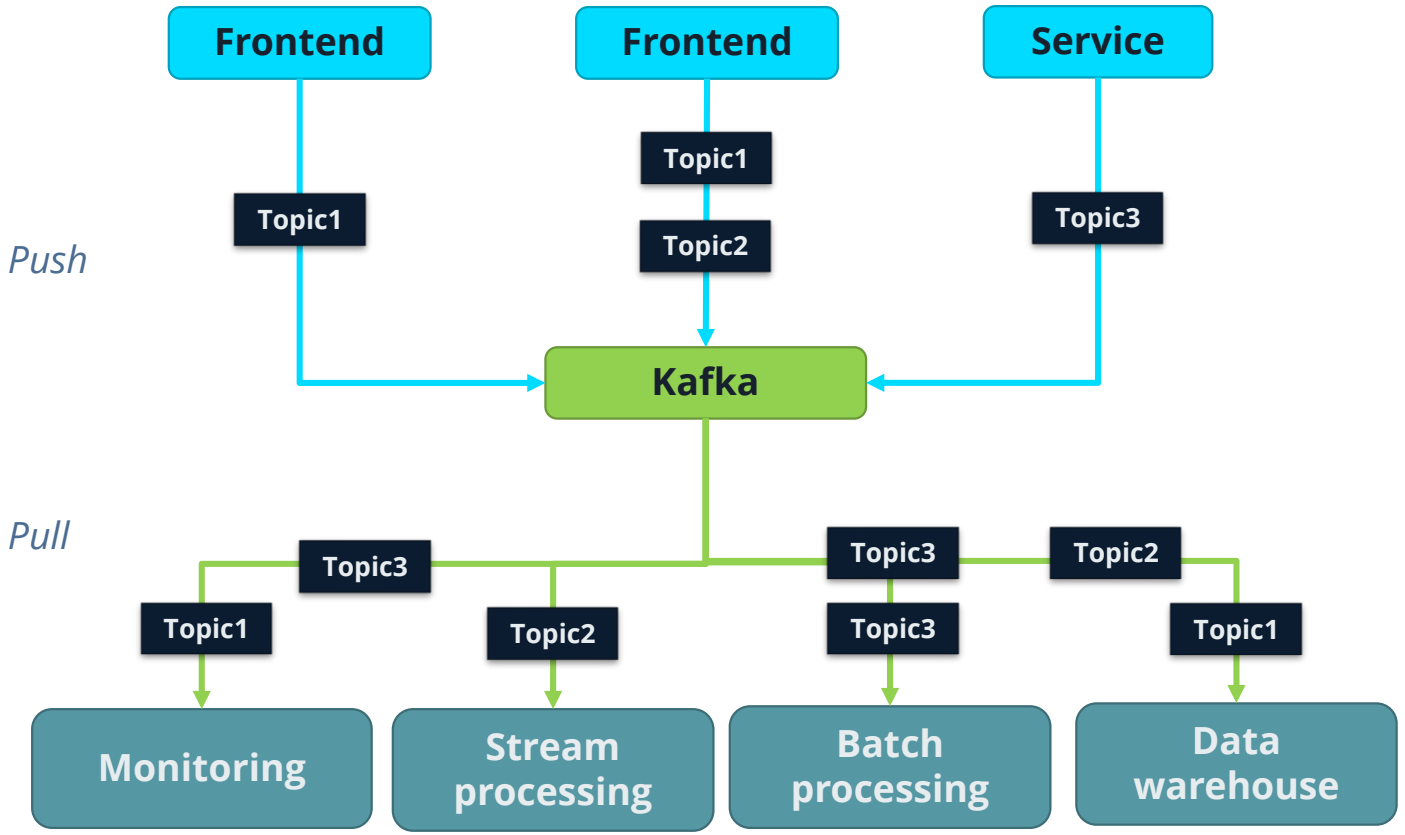


CONCEPTOS BÁSICOS

Productores

Broker

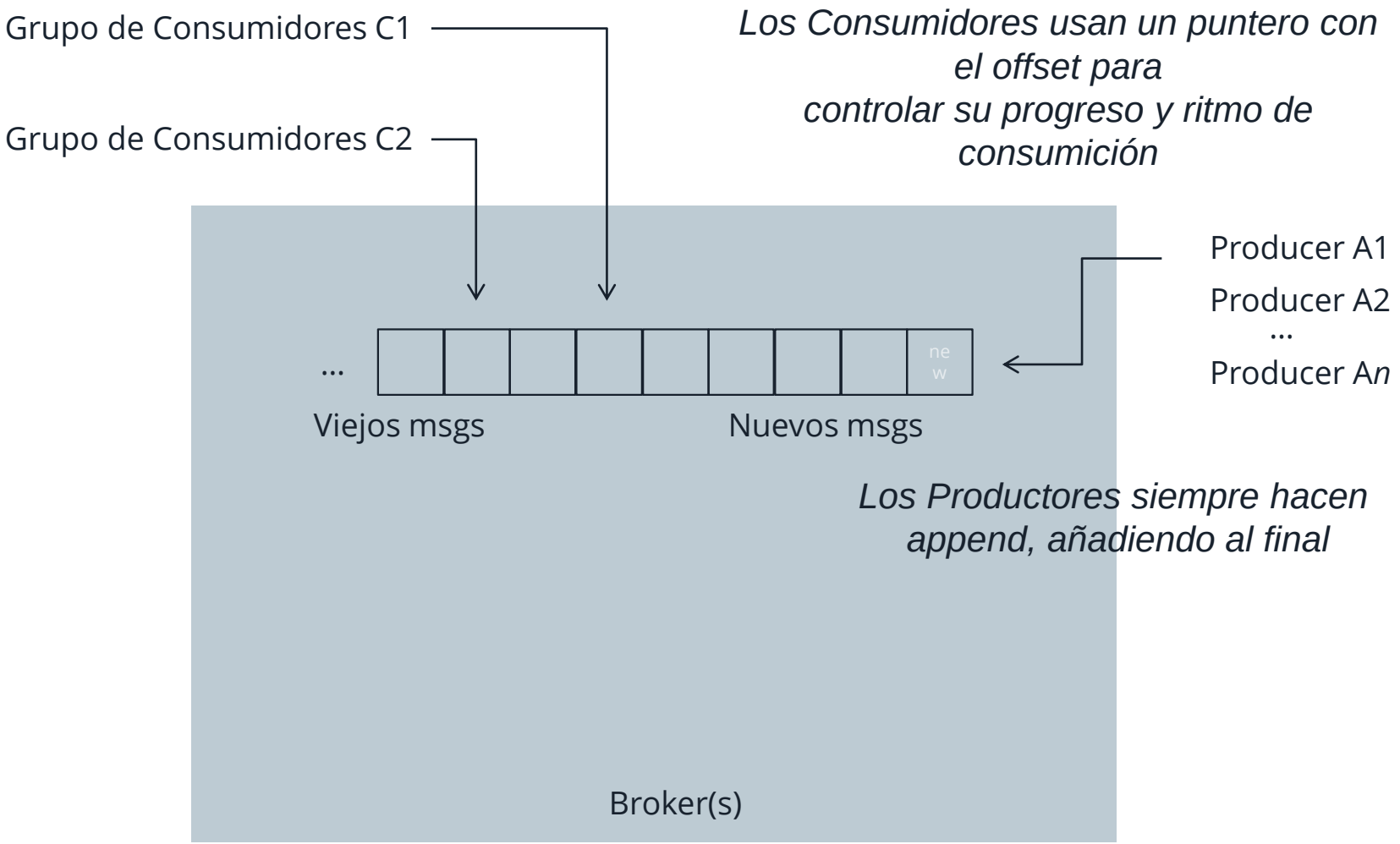
Consumidores



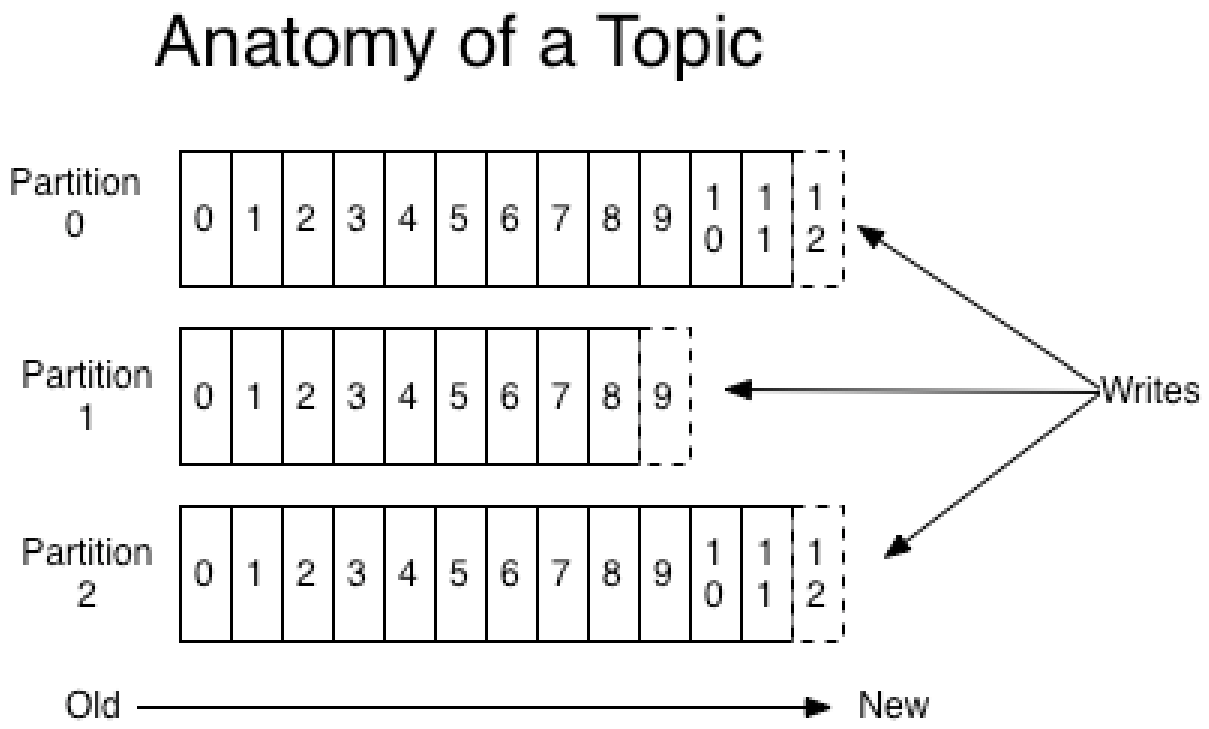
CONCEPTOS

- **Topic:** flujo donde se publican los mensajes
- Un **Topic** se compone de particiones
  - **Partición**: secuencia de mensajes **ordenada** e **inmutable** a la que se añaden mensajes por el final
- **Offset**: ID único (por partición) y secuencial que tienen asignado los mensajes en una partición
  - Los Consumidores pueden manejar punteros en base a tuplas *(offset, partition, topic)*
- **Replicas:** “backups” de una partición
  - Su propósito es evitar perdidas de datos

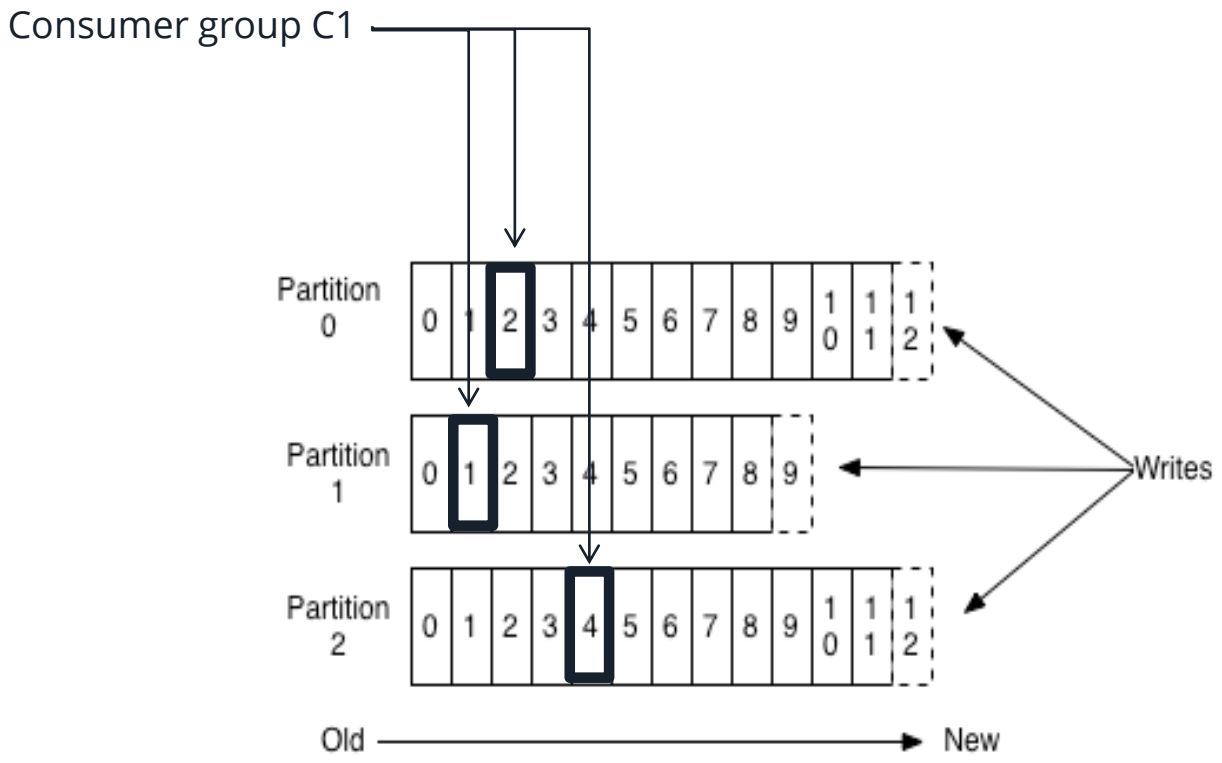
TOPICS


Grupo de Consumidores C1
Grupo de Consumidores C2
Los Consumidores usan un puntero con el offset para controlar su progreso y ritmo de consumición
Producer A1
Producer A2
...
Producer An
...
new
Viejos msgs
Nuevos msgs
Los Productores siempre hacen append, añadiendo al final
Broker(s)

PARTICIONES

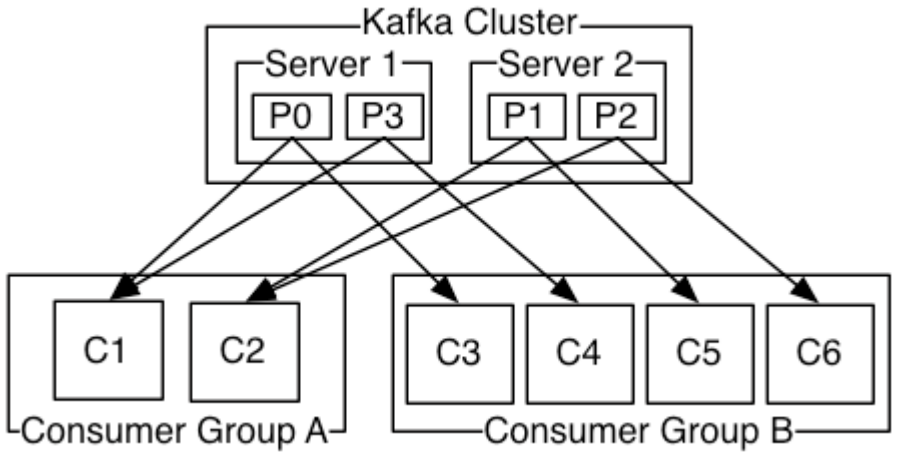


OFFSETS


Consumer group C1
Partition 0
0 1 2 3 4 5 6 7 8 9 10 11 12
Partition 1
0 1 2 3 4 5 6 7 8 9
Partition 2
0 1 2 3 4 5 6 7 8 9 10 11 12
Writes
Old
New

PARTICIONES

- El número de Particiones de un Topic es configurable
- Ese número de Particiones determina el paralelismo máximo
  - Grupo A, con dos Consumidores, lee de un Topic con 4 Particiones
  - Grupo B, con cuatro Consumidores, lee del mismo Topic

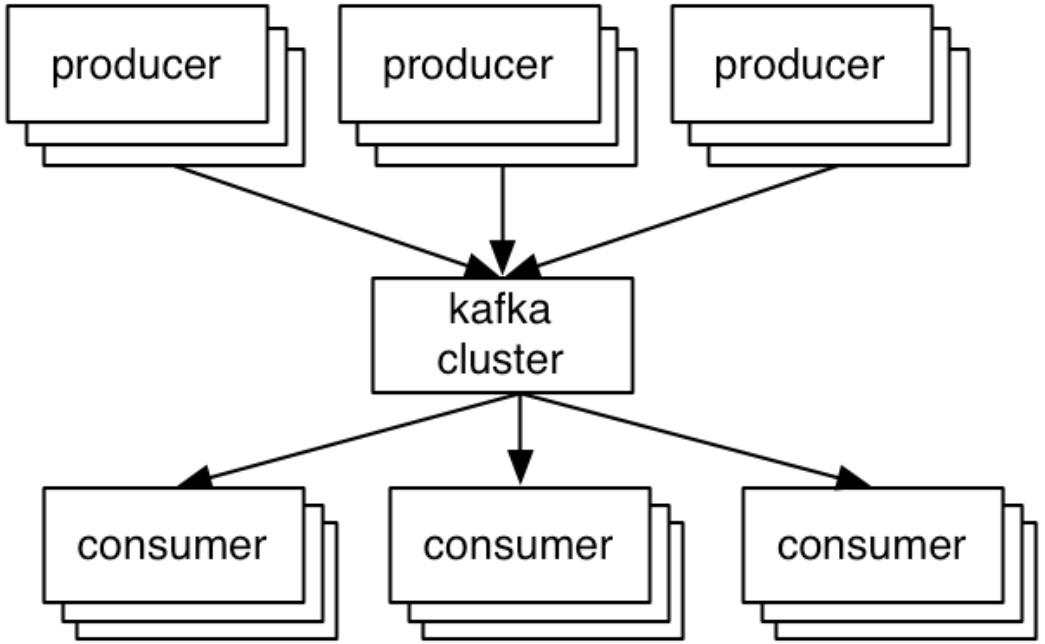


REPLICAS

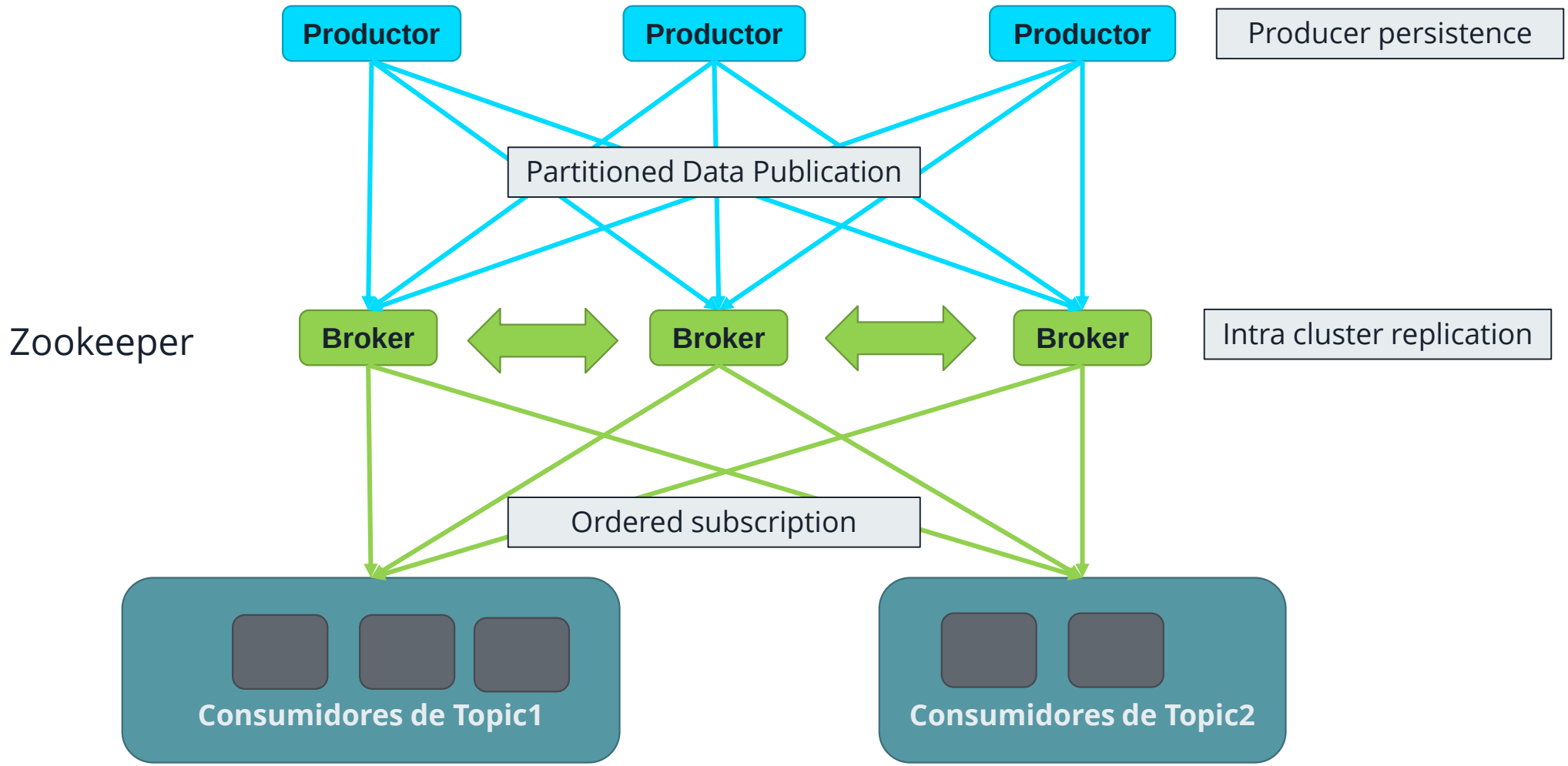
- Replicas: “backups” de una Partición
  - Su único propósito es evitar la perdida de datos
  - No hay lecturas ni escrituras en las Replicas
    - No aportan paralelismo al Productor ni al Consumidor
  - Kafka permite perder (numReplicas - 1) Brokers antes de que se produzca una pérdida de datos
    - numReplicas == 2 → Podemos permitirnos perder 1 Broker

ARQUITECTURA BÁSICA

- Los actores
  - Los **Productores** escriben en los **Brokers**
  - Los **Consumidores** leen de los **Brokers**
  - Todo en una arquitectura distribuida
- Los datos
  - Los datos se almacenan en **Topics**
  - Los **Topics** se separan en **Particiones**, que están replicadas



MODELO DISTRIBUIDO



CASOS DE USO

Stream Processing

Messaging

Click Streams

Metrics Collection

Log Aggregation

DESVENTAJAS

- No esta optimizado para latencias de ms
- Sistema simple de mensajes, sin procesado
- Zookeeper puede ser un problema con muchos Topics/Particiones
  - Mas de 10000
- No esta diseñado para mensajes de gran tamaño

plain concepts

APACHE KAFKA

plain concepts

STORM Y KAFKA

CONFIABILIDAD DE LAS FUENTES DE DATOS

- Una fuente de datos se considera ***no-fiable (unreliable)*** si no hay modo de recuperar un mensaje recibido previamente
- Una fuente de datos se considera ***fiable (reliable)*** si hay un modo de recuperar un mensaje en caso de que el procesado falle en cualquier momento
- Una fuente de datos se considera ***duradera (durable)*** si hay modo de recuperar un mensaje o un conjunto de los mismos en base a unos criterios de selección

CONFIABILIDAD EN STORM

- Procesar un mensaje **exactamente** una vez requiere una fuente de datos ***duradera***
- Procesar un mensaje **al menos** una vez requiere una fuente de datos ***fiable***
- Una fuente de datos ***no-fiable*** puede (y debe) apoyarse de algún modo para proporcionar una mínima garantía
- Con fuentes de datos ***duradera*** o ***fiable*** Storm no perderá datos
- Podemos apoyar una fuente de datos ***no-fiable*** con Apache Kafka
  - Pequeña disminución del rendimiento a cambio de una fuente de datos ***duradera***

STORM Y KAFKA

- Apache Kafka es una Fuente de datos ideal para una topología Storm ya que proporciona los mecanismos para garantizar que el mensaje se procese
  - Al menos una vez
  - Como mucho una vez
  - Exactamente una vez
- Apache Storm incluye spouts de Kafka
- Kafka soporta una amplia variedad de lenguajes e integraciones a nivel de Productor y de Consumidor
- En nuestros ejemplos de Storm no hemos usado Kafka sino EventHub

plain concepts

KAFKA VS EVENT HUB

KAFKA VS EVENT HUB I

- Kafka
  - IaaS
  - Opción on-premises
  - No soporta AMQP
  - Principalmente Java/Scala
  - No es sencillo usar geo-replicación
  - Sin throttling
- Event Hub
  - PaaS
  - No hay opción on-premises
  - Soporta AMQP
  - C#/.NET
  - Sencillo usar geo-replicación
  - Throttling (TUs)

KAFKA VS EVENT HUB II

- Kafka
  - Menos opciones de seguridad
  - Integración compleja
  - Sin máximo tamaño mensaje
  - Rendimiento
- Event Hub
  - Seguridad Azure
  - Integración entorno Azure
  - Mensaje máximo 256 KB
  - Rendimiento

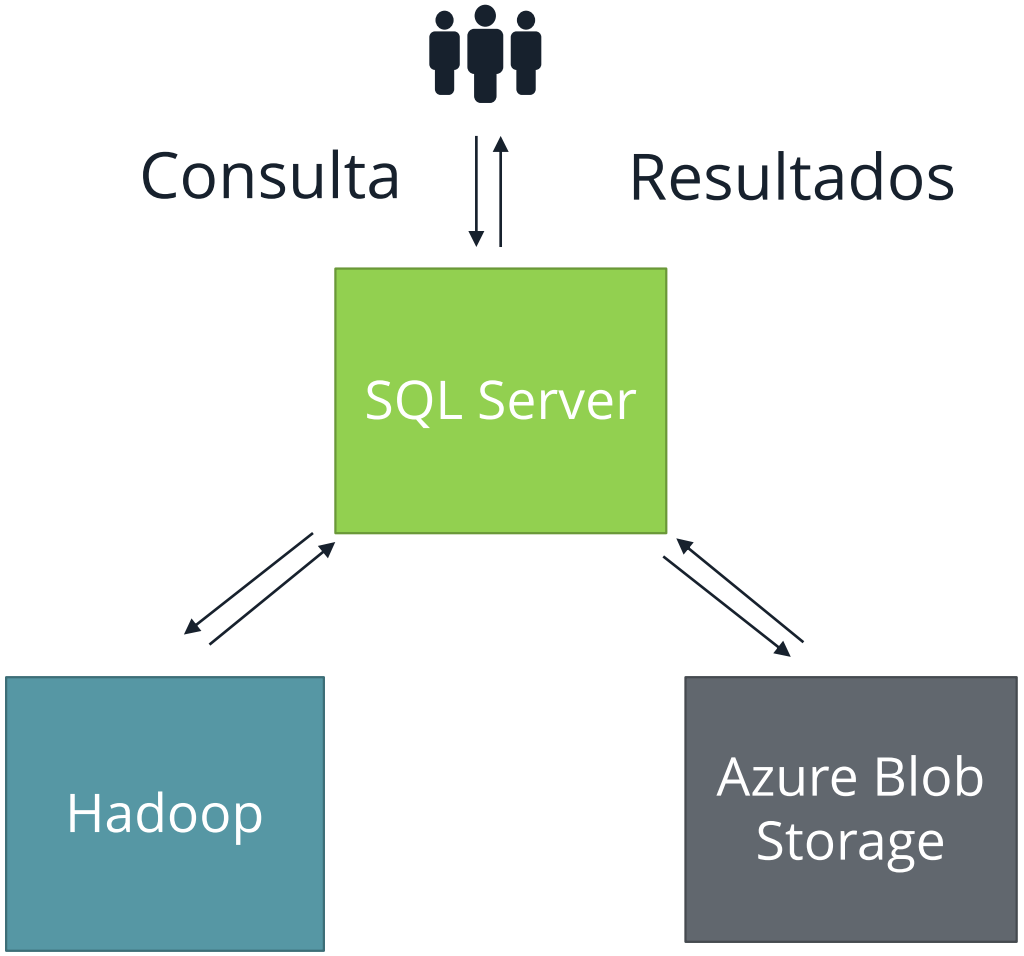
POLYBASE

plain concepts

POLYBASE

PolyBase proporciona una extensión de T_SQL para combinar datos en Hadoop y en un RDMS

POLYBASE EN SQL SERVER 2016



POLYBASE USE CASES

Carga de datos

Uso de Hadoop como una herramienta más de ETL para preparar los datos antes de cargarlos en nuestro DW utilizando PolyBase

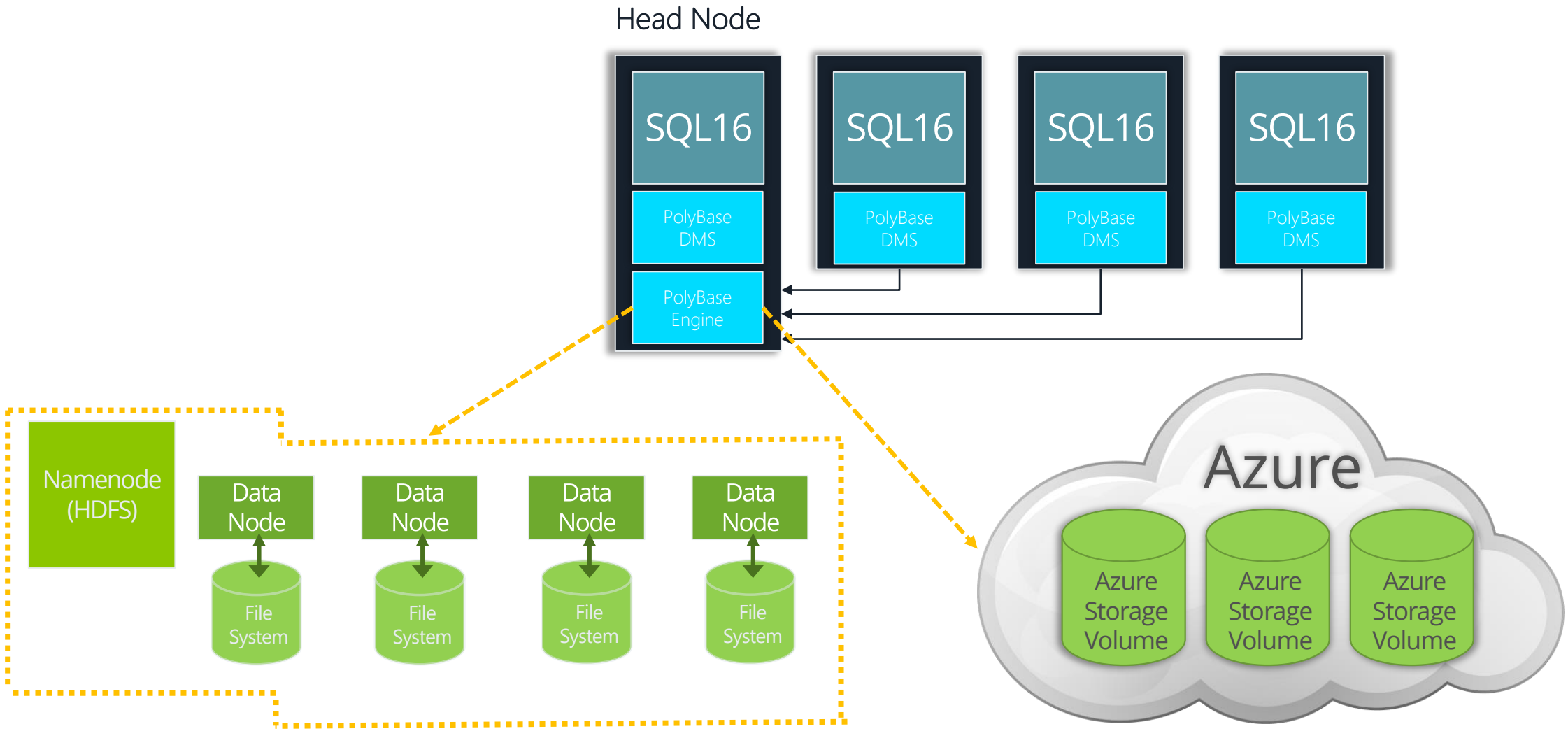
Consultas interactivas

Combinar y analizar datos relacionales y datos semi estructurados

Almacenamiento frio

Enviar datos antiguos a HDFS o Azure Storage para mantenerlos en un almacenamiento frio, mas barato pero aun consultable

ARQUITECTURA



FUNCIONAMIENTO DE POLYBASE

Lectura/escritura en varios formatos

e.g. Text, RCFILE, ORC, Parquet, Avro

Paralelización de las transferencias

entre el DW y los nodos HDFS

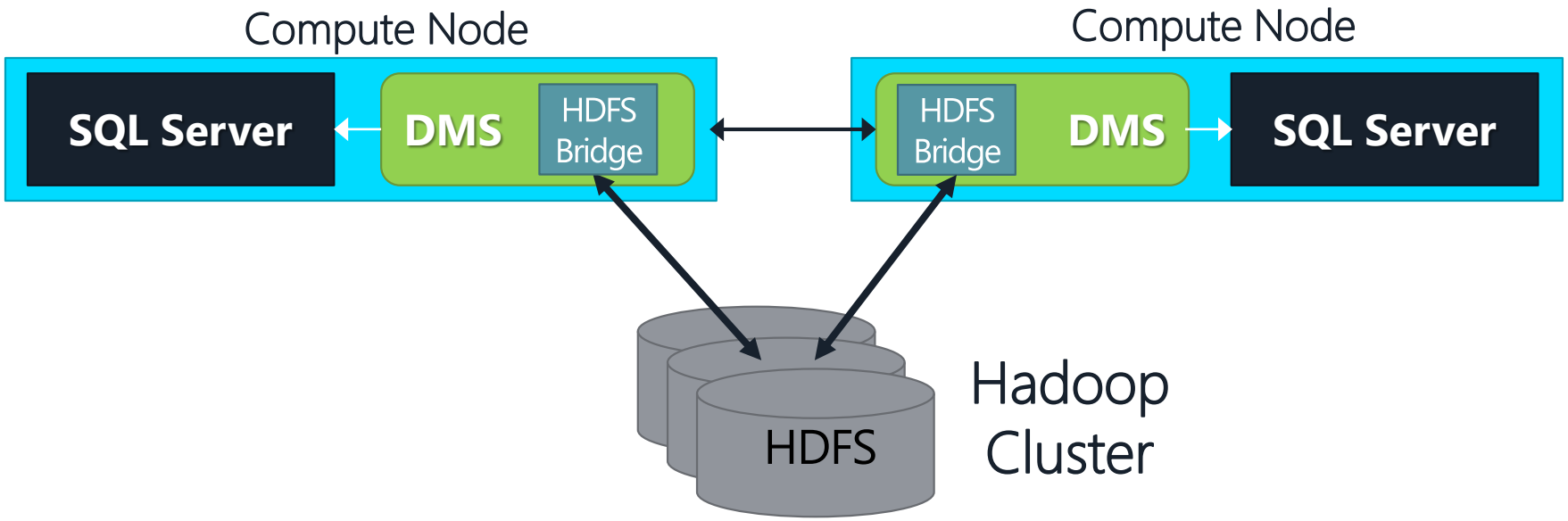
Dando estructura a datos semi-estructurados

mediante el concepto de external table

Aprovechando los clústeres Hadoop para la computación

con push-down computation

LECTURA Y ESCRITURA



Utiliza los RecordReaders/RecordWriters de Hadoop para leer/escribir formatos de fichero standard de HDFS

FUNCIONAMIENTO DE POLYBASE

Lectura/escritura en varios formatos

e.g. Text, RCFILE, ORC, Parquet, Avro

Paralelización de las transferencias

entre el DW y los nodos HDFS

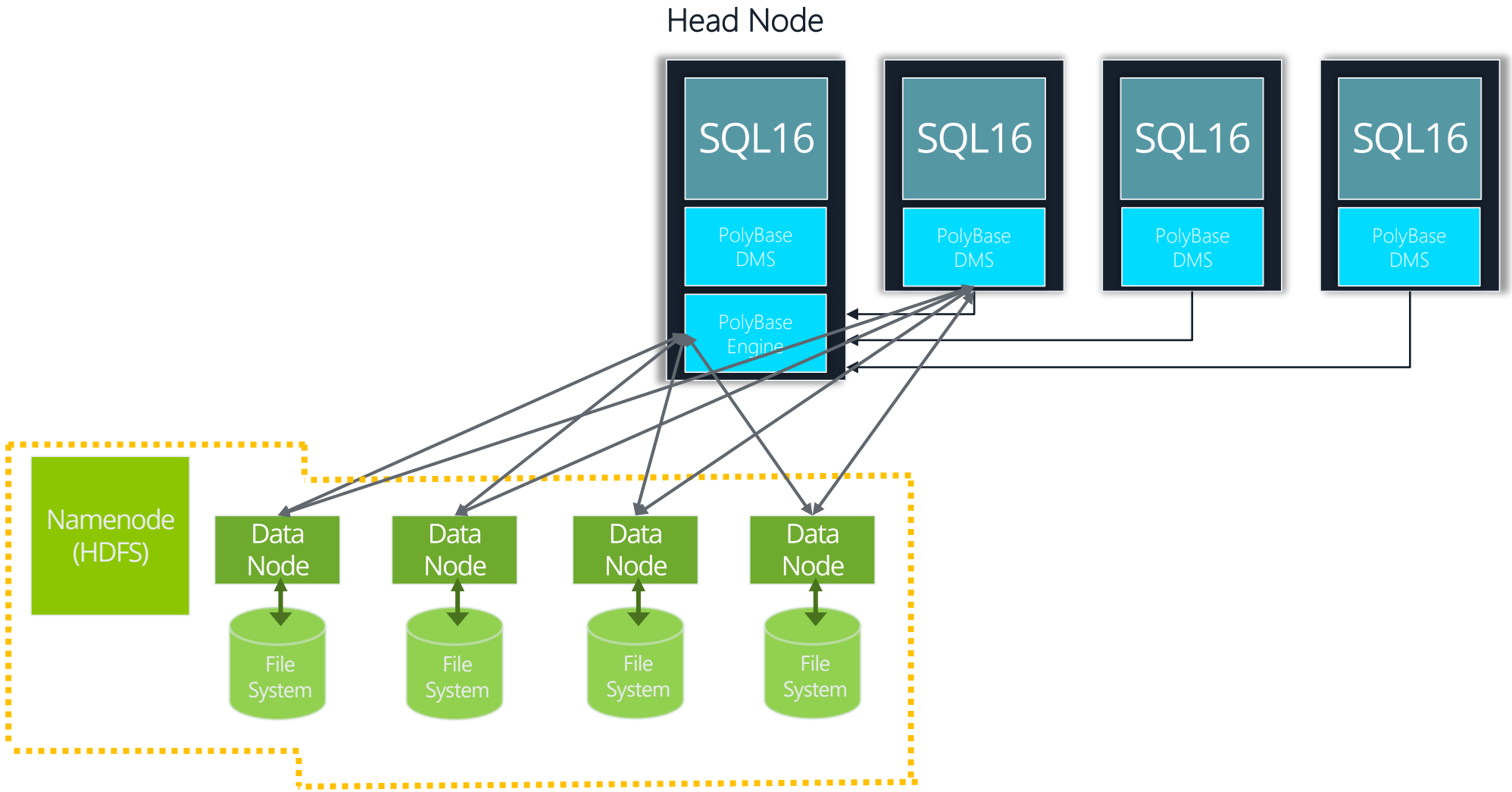
Dando estructura a datos semi-estructurados

mediante el concepto de external table

Aprovechando los clústeres Hadoop para la computación

con push-down computation

PARALELIZACIÓN DE LAS TRANSFERENCIAS



FUNCIONAMIENTO DE POLYBASE

Lectura/escritura en varios formatos

e.g. Text, RCFILE, ORC, Parquet, Avro

Paralelización de las transferencias

entre el DW y los nodos HDFS

Dando estructura a datos semi-estructurados

mediante el concepto de external table

Aprovechando los clústeres Hadoop para la computación

con push-down computation

TABLAS EXTERNAS

```
CREATE EXTERNAL DATA SOURCE MyHadoopCluster
WITH (TYPE = Hadoop, LOCATION = 'hdfs://10.193.26.177:8020',
                  RESOURCE_MANAGER_LOCATION = '10.193.26.178:8050');

CREATE EXTERNAL FILE FORMAT MyTextFile
WITH ( FORMAT_TYPE = DELIMITEDTEXT,
                  DATA_COMPRESSION = 'org.apache.hadoop.io.compress.GzipCodec',
                  FORMAT_OPTIONS (FIELD_TERMINATOR ='|');

CREATE EXTERNAL TABLE [dbo].[SensorData] (
    [SensorKey] int NOT NULL,
    [CustomerKey] int NOT NULL,
    [Speed] float NOT NULL
)
WITH (LOCATION='//Sensor_Data//May2014/sensordata.tbl',
      DATA_SOURCE = MyHadoopCluster,
      FILE_FORMAT = MyTextFile
);
```

Una vez por cluster

Una vez por formato

HDFS File Path

TABLAS EXTERNAS (SECURIZADAS)

```
CREATE DATABASE SCOPED CREDENTIAL HadoopCredential
WITH IDENTITY = 'hadoopUserName', Secret = 'hadoopPassword';
```

Una vez por usuario

```
CREATE EXTERNAL DATA SOURCE MyHadoopCluster
WITH (TYPE = Hadoop, LOCATION = 'hdfs://10.193.26.177:8020',
                        RESOURCE_MANAGER_LOCATION = '10.193.26.178:8050',
                        CREDENTIAL = HadoopCredential);
```

Una vez por usuario

```
CREATE EXTERNAL FILE FORMAT MyTextFile
WITH ( FORMAT_TYPE = DELIMITEDTEXT,
                        DATA_COMPRESSION = 'org.apache.hadoop.io.compress.GzipCodec',
                        FORMAT_OPTIONS (FIELD_TERMINATOR ='|', USE_TYPE_DEFAULT = TRUE));
```

Una vez por formato

```
CREATE EXTERNAL TABLE [dbo].[SensorData] (
    [SensorKey] int NOT NULL,
    [CustomerKey] int NOT NULL,
    [Speed] float NOT NULL
)
WITH (LOCATION='//Sensor_Data//May2014/',
      DATA_SOURCE = MyHadoopCluster,
      FILE_FORMAT = MyTextFile
);
```

HDFS File Path

FUNCIONAMIENTO DE POLYBASE

Lectura/escritura en varios formatos

e.g. Text, RCFILE, ORC, Parquet, Avro

Paralelización de las transferencias

entre el DW y los nodos HDFS

Dando estructura a datos semi-estructurados

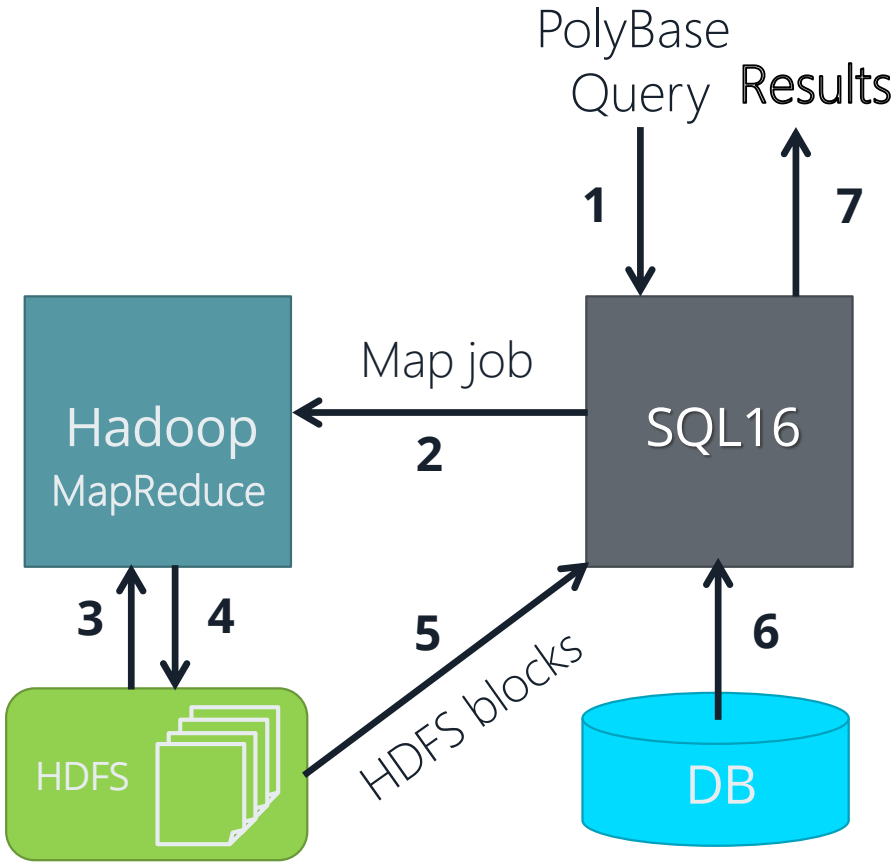
mediante el concepto de external table

Aprovechando los clústeres Hadoop para la computación

con push-down computation

PUSH-DOWN COMPUTATION

Operaciones SQL sobre datos HDFS se convierten en trabajos MapReduce



Decisión en base al coste de cuanta computación llevarnos a Hadoop

plain concepts

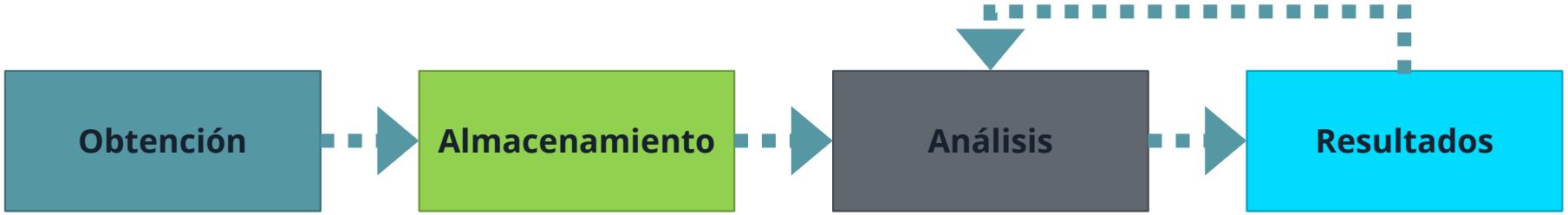
POLYBASE

AZURE DATA LAKE

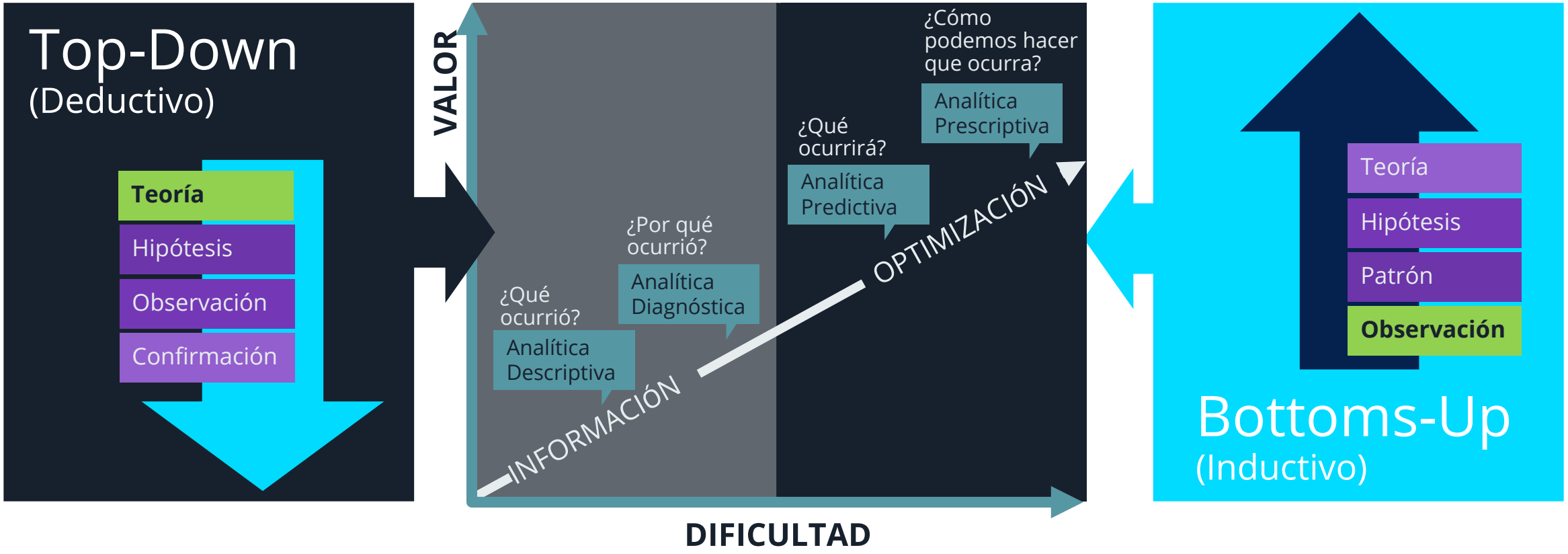
plain concepts

TODOS LOS DATOS TIENEN VALOR

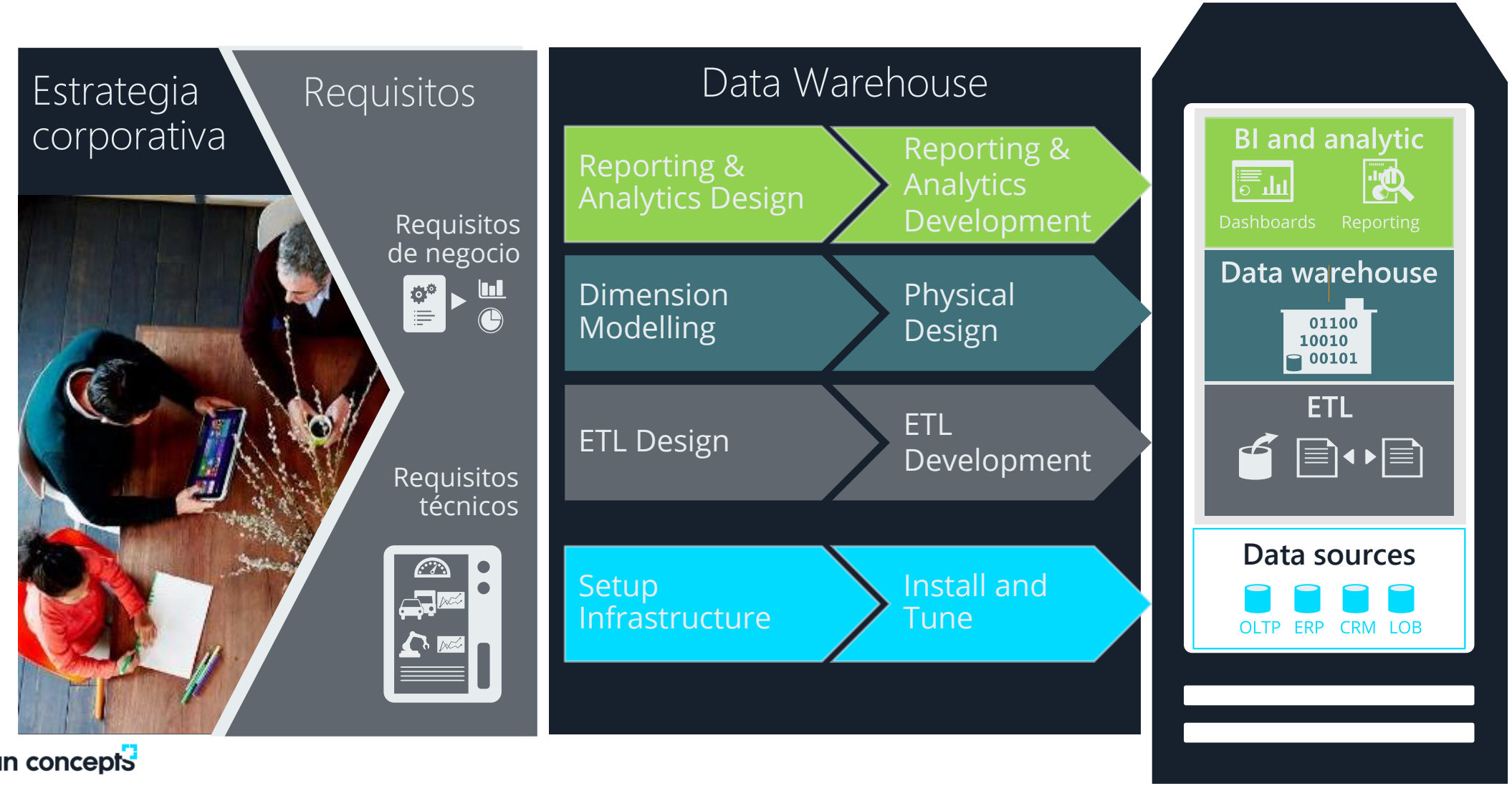
- Todos los datos tienen valor - Data hoarding
- Esquema indefinido, almacenamos datos en formato nativo
  - Schema on read
- Interpretamos los datos cuando nos interesa



TOP DOWN VS. BOTTOMS UP



DATA WAREHOUSING TRABAJA EN TOP-DOWN

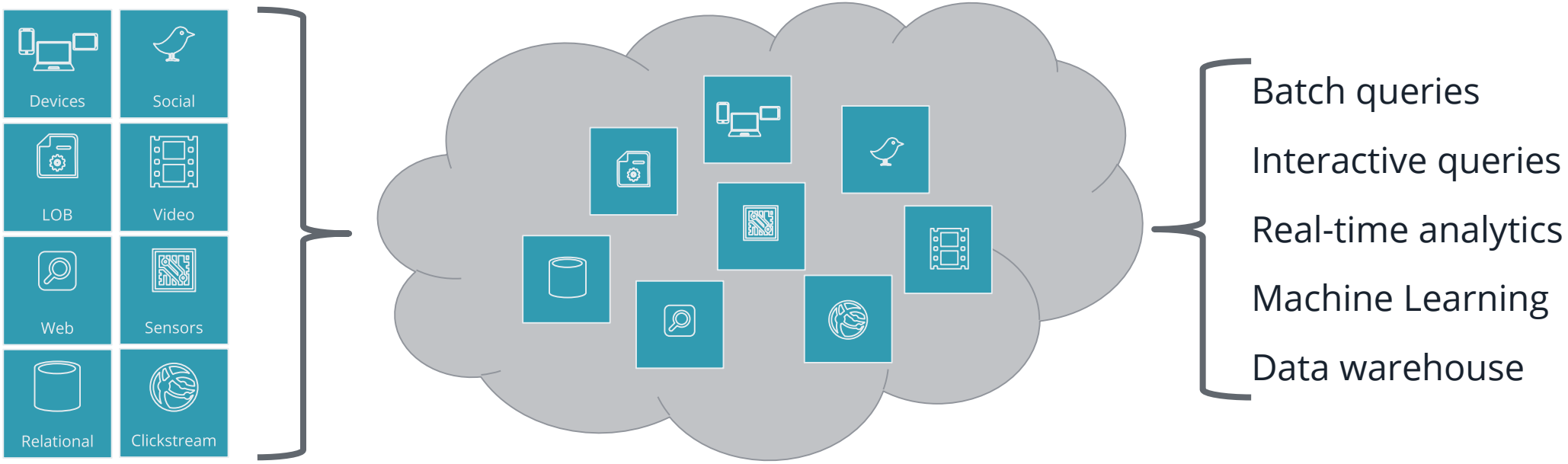


DATA LAKE TRABAJA EN BOTTOMS-UP

Ingerir
todos los datos, sin importar los requisitos

Almacenar
en formato nativo, sin esquema definido

Analizar
usando un motor analítico como Hadoop



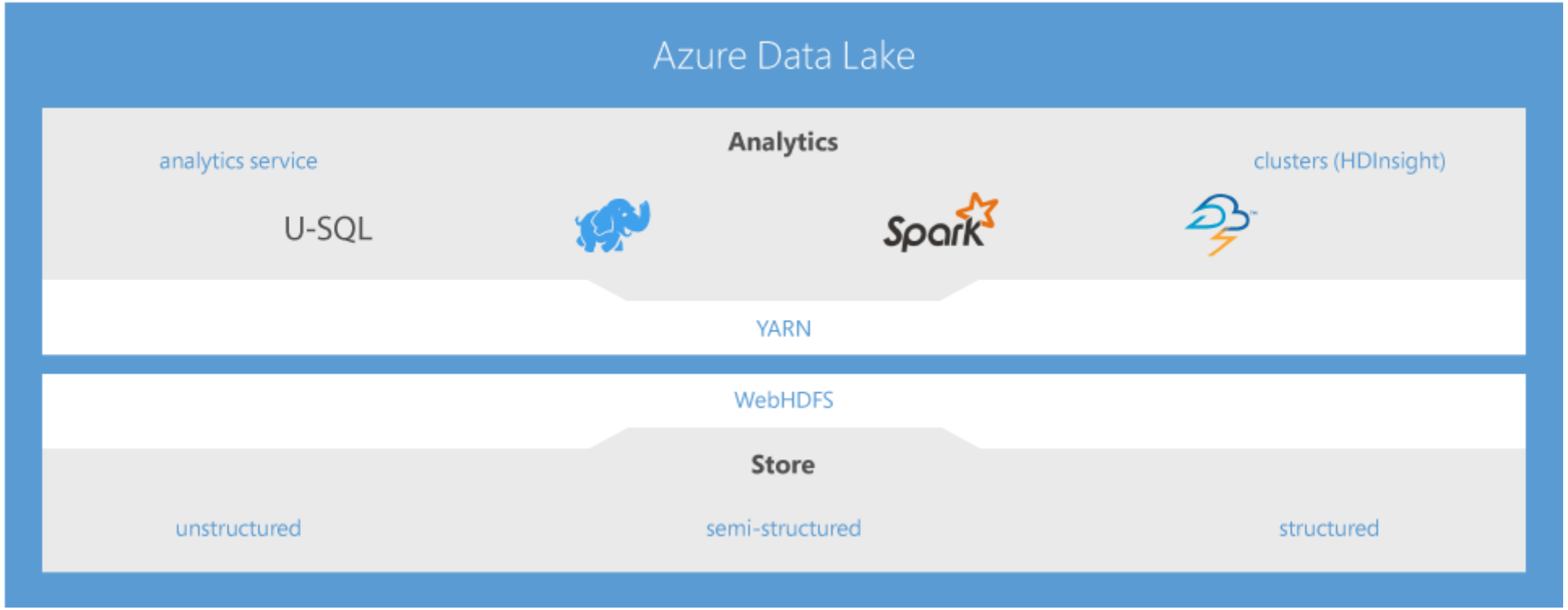
AZURE DATA LAKE

- Data Lake Store es un repositorio de datos en su formato original
  - Diseñado para volúmenes infinitos y alto rendimiento en el procesado y el análisis
  - Gran cantidad y variedad, rapidez, estructurados y no estructurados...lo que viene siendo Big Data
- Azure Data Lake Analytics es un servicio analítico distribuido construido sobre YARN
  - Permite ejecutar jobs en U-SQL directamente contra los datos almacenados en el Data Lake
- Azure Data Lake HDI es HDInsight sobre Azure Data Lake
  - Las mismas características de HDInsight, pero trabajando con datos almacenados en Data Lake

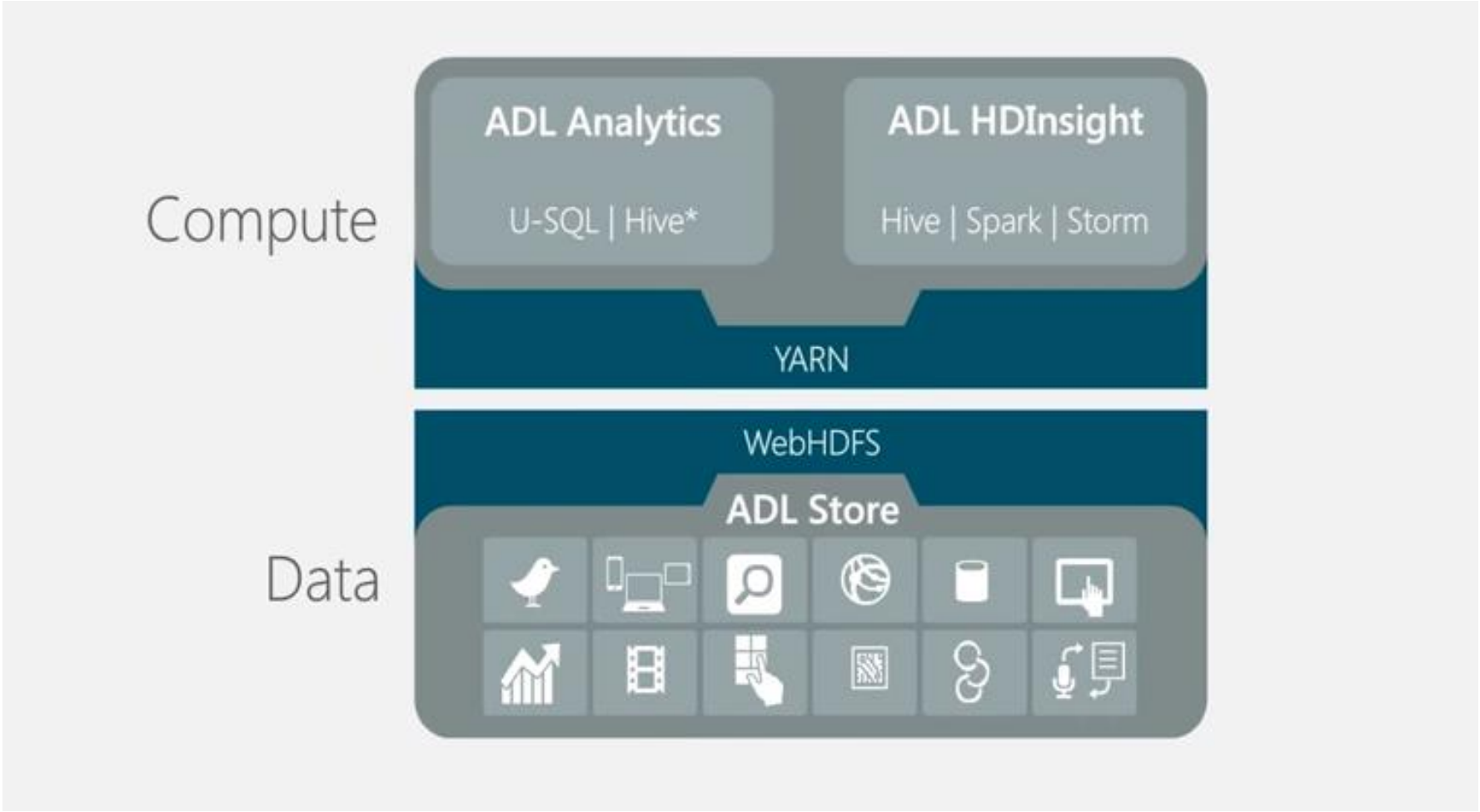
AZURE DATA LAKE

- Almacenamiento y análisis de datos de cualquier tipo y tamaño
- Exploración interactiva de patrones en los datos
- U-SQL, Apache Spark, Hive, HBase, Storm...
- Fácilmente escalable dinámicamente
- Integrado con Azure Active Directory
- Basado en YARN

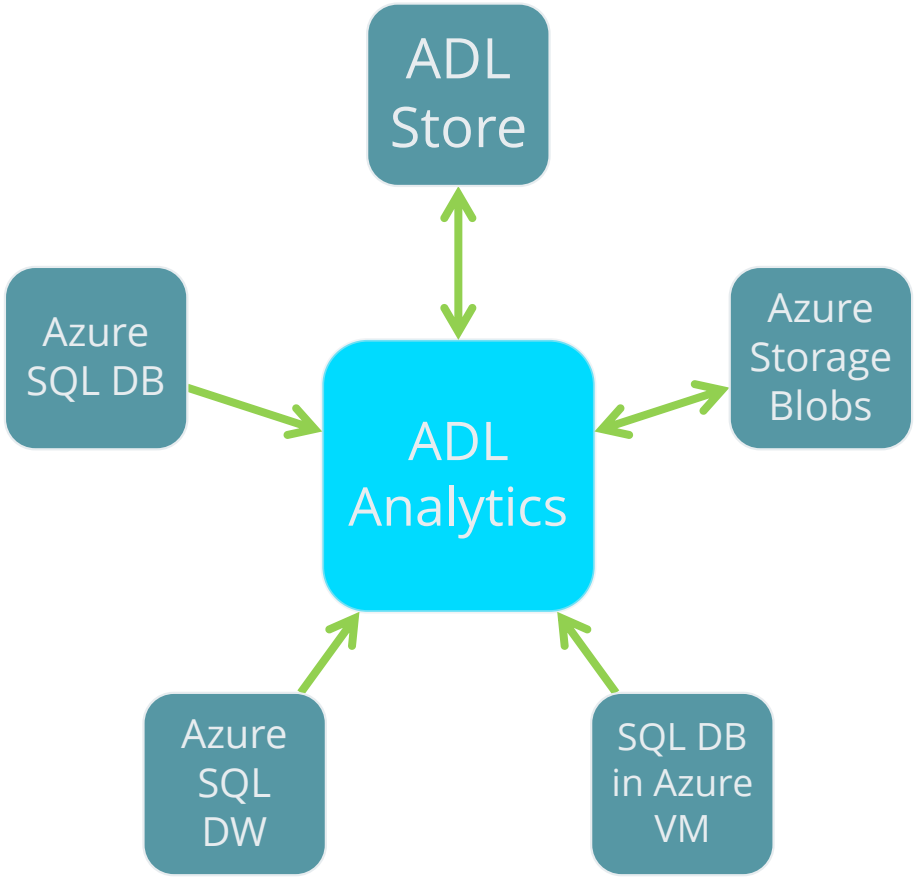
DATA LAKE


Azure Data Lake
Analytics
analytics service
U-SQL
Spark
clusters (HDInsight)
YARN
WebHDFS
Store
unstructured
semi-structured
structured

DATA LAKE


Compute
ADL Analytics
U-SQL | Hive*
ADL HDInsight
Hive | Spark | Storm
YARN
Data
WebHDFS
ADL Store

DATA LAKE ANALYTICS



U-SQL

- Lenguaje de proceso que une la naturaleza declarativa de SQL con la potencia de C#
  - Sistema de metadatos, sintaxis y semántica SQL
  - Tipos de datos y expresiones C#
  - Integra características de Hive para proceso de datos

plain concepts

AZURE DATA LAKE

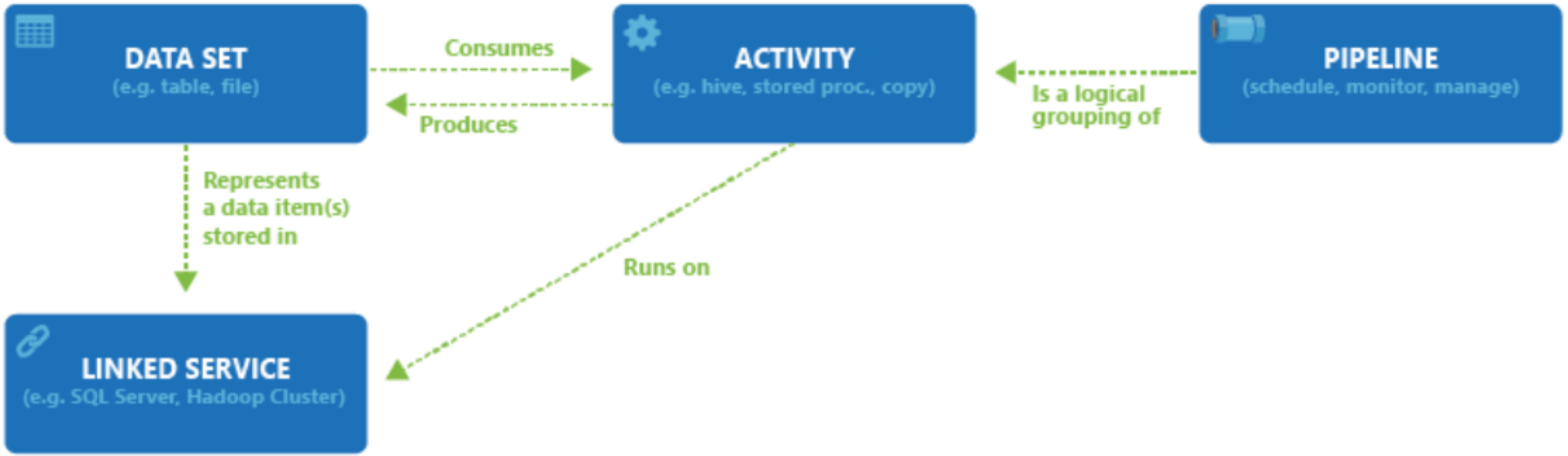
AZURE DATA FACTORY

plain concepts

AZURE DATA FACTORY

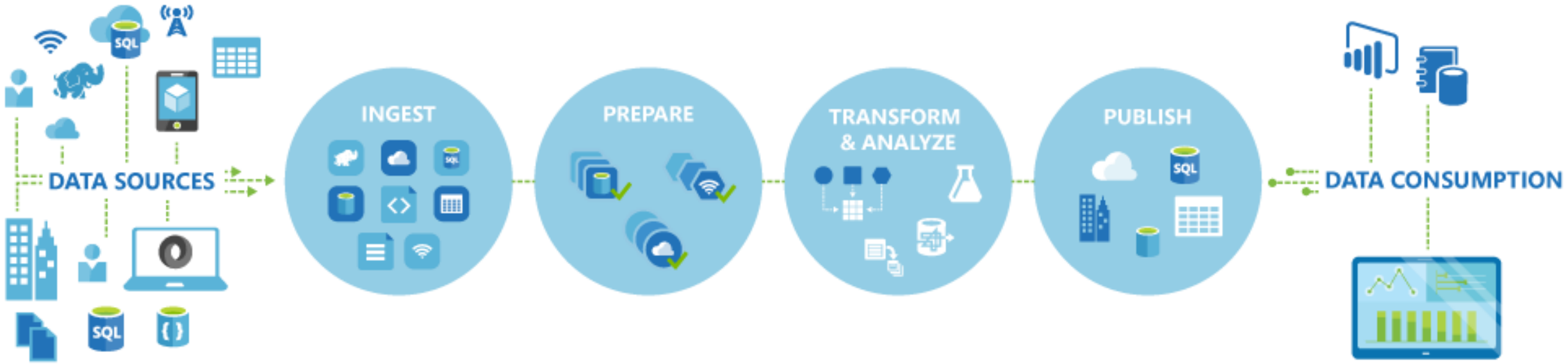
- Azure Data Factory es un Sistema de integración de datos en la nube
  - Automatiza el movimiento y la transformación de los datos
- Compone el procesado, el almacenamiento y el movimiento de datos para generar una pipeline de análisis
- Ofrece un Sistema completo de monitorización y gestión del proceso
  - De principio a fin
- Inicialmente pensado para Azure y trabajo con datos en un SQL Server on Premises
  - Data Management Gateway

COMPONENTES ADF



FLUJO DE DATOS

Overview diagram



LINKED SERVICES

- Los Linked Services definen la conexión de ADF a recursos externos
- Se utilizan para representar
  - Una fuente de datos, incluyendo: Azure Storage, Azure SQL, Azure SQL Data Warehouse, Azure DocumentDB, SQL Server, Oracle, File System, DB2, MySQL, Teradata, PostgreSQL, Sybase
  - Un recurso de computación que ejecuta una actividad
    - “HDInsightHiveActivity” se ejecuta en HDI

DATASETS

- Los Datasets son referencias con nombre a un conjunto de datos asociado con una Activity
  - Pueden ser de entrada o de salida
- Identifican estructuras dentro de una fuente de datos
  - Tablas, filas, directorios, documentos...

ACTIVITIES

- Una Activity define acciones a realizar sobre un conjunto de datos
- Cada Activity toma cero o más Datasets de entrada y produce uno o más Datasets de salida
- Es la principal unidad de orquestación de datos de ADF

ACTIVITIES

- Data Movement Activities
  - Copy

- Transformation Activities
  - Hive (HDI)
  - Pig (HDI)
  - MapReduce (HDI)
  - Hadoop Streaming (HDI)
  - Machine Learning (Azure VM)
  - Procedimiento Almacenado (Azure SQL)
  - Código .NET (HDI o Azure Batch)

PIPELINES

- Las Pipelines representan una agrupación lógica de Activities
- Sirven para agrupar Activities en una unidad que realiza una tarea
- Las Activities agrupadas en un Pipeline pueden desplegarse, programarse borrarse como una unidad

PROCESO

- Definición de la arquitectura
- Creación de Data Factory
- Creación de Linked Services
- Creación de Datasets de entrada y salida
- Creación del Pipeline
  - Definición de actividades
- Monitorización

DATA MANAGEMENT GATEWAY

- Data Management Gateway permite un acceso seguro entre Azure y una fuente de datos on-premises
  - Sin cambios en el firewall (trabaja con conexiones HTTP)
  - Encriptando las credenciales con el certificado del cliente
  - Transfiriendo datos en paralelo y con sistemas de reintento automáticos
- Cada DMG trabaja con una instancia de ADF
- Solo un DMG se puede instalar en una máquina

plain concepts

AZURE DATA FACTORY

¿PREGUNTAS?

plain concepts

plain concepts

EJERCICIO PRÁCTICO

GRACIAS

plain concepts

Barcelona

Bilbao

Madrid

Sevilla

Dubai

London

Seattle

plain concepts