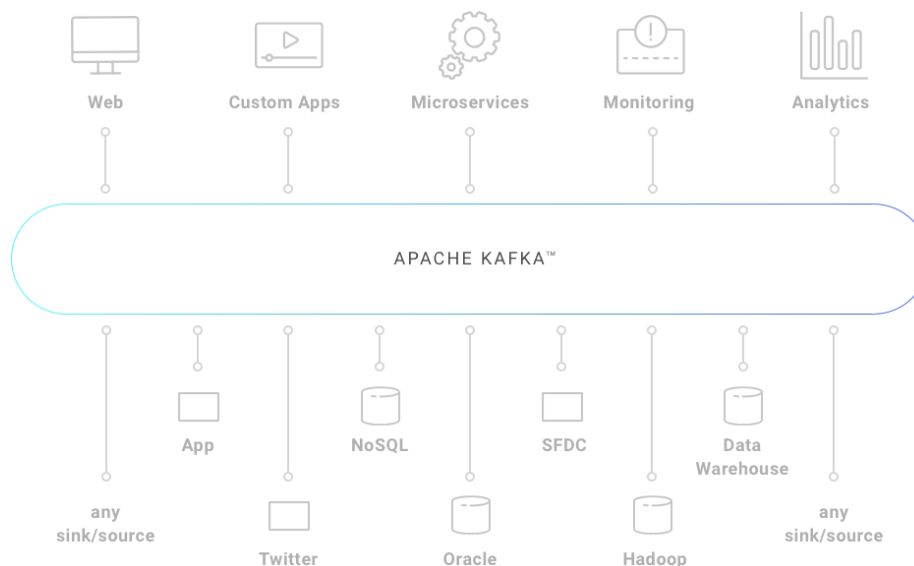


Apache Kafka - PADSA

Apache Kafka es una reconocida plataforma de transmisión de eventos que se usa para recopilar, procesar y almacenar datos de eventos de transmisión o datos que no tienen un principio y un final distinguidos. Kafka posibilita una nueva generación de aplicaciones distribuidas con la capacidad de escalar para controlar miles de millones de eventos transmitidos por minuto.

Fue creado en 2010 en LinkedIn para el rastreo de comportamientos de usuarios en el portal.

Kafka tiene por objetivo crear un gran bitácora de eventos altamente eficiente para almacenar lo que esta sucediendo y mantenerlos disponibles para procesar el flujo de información. (Kafka esta pensado para aplicaciones de altos volúmenes de información constante para ser analizados en tiempo real).



Facilita mucho la integración en tu organización por la estructura de su sistema, dividiendo cada una de sus funcionalidades en una API: Producer, Consumer, Streams y Connect.

- **Producer API**, que te permite que una aplicación publique una secuencia de *records* a uno o más *topics* de Kafka.
- **Consumer API** permite que una aplicación se suscriba a uno o más *topics* y procese la secuencia de *records* disponibles.
- **Streams API** permite que una aplicación actúe como un *procesador de flujo de datos*, consumiendo un flujo de entrada de uno o más *topics* y produciendo un flujo de salida a uno o más *topics* de salida, transformando los flujos de entrada en flujos de salida diferentes.
- **Connect API** permite crear y ejecutar *producer* o *consumer* reutilizables que conectan los *topics* de Kafka a las aplicaciones de tu organización. Por ejemplo, un conector a una base de datos **MySQL**.

Características

- Es altamente escalable
- Tolerancia a fallos
- Manejo de datos en tiempo real
- Existen productores y consumidores
- Posee 2 modelos: Publicador/Suscriptor y Cola de mensajes
- Alta transaccionalidad y baja latencia por el manejo de replicación y distribución en nodos
- Kafka puede manejar una Arquitectura para trabajar bajo eventos
- Los datos se almacenan en la arquitectura de apache kafka, la información se escribe en discos en forma de logs.

Conceptos Básicos

Mensaje: datos que se envían para ser almacenados o consumidos.

Tópico: Categoría o criterio con el que se pueden agrupar/clasificar los mensajes en Kafka. Una colección de mensajes conforman un topic.

Pueden existir 1 o más topics -> No hay limitación, Kafka almacena los topics en logs

Productor: encargado de publicar los mensajes en los tópicos.

Consumidor: encargado de obtener los mensajes de los tópicos y procesarlos.

Broker: es una instanciación de un servidor de kafka.

Clúster: Agrupación de 1 o N Brokers

Esquema: Estructura opcional que se aplica sobre los mensajes garantizando una distribución concreta de la información. *Por ejemplo : JSON , XML y Apache Avro*

Partition / Partición: Secuencia de mensajes ordenada e inmutable. Cada uno de los partes en las que se puede fragmentar/dividir el "Topic Log" de un topic.

Replica: La cantidad de veces que un *topic* se replicara en los nodos de kafka creados.

Offset: También se denomina "desplazamiento" o "compensación" Kafka guarda su información en un *topic* llamado "_consumer_offset"

Mensaje

Unidad de datos con la que trabaja Kafka (sería la unidad de comunicaciones entre aplicaciones)

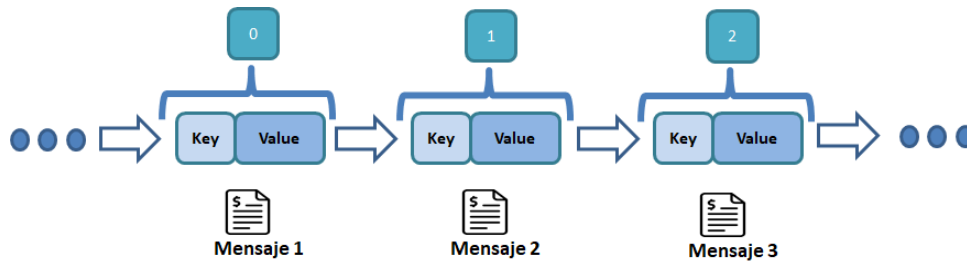
| Un mensaje sería similar a un registro de base de datos

Características

- También se usa el término "registro" / "record" / "message"
- No tiene una estructura, formato o significado específico para Kafka
- Se persiste dentro de una partición que pertenece a un topic

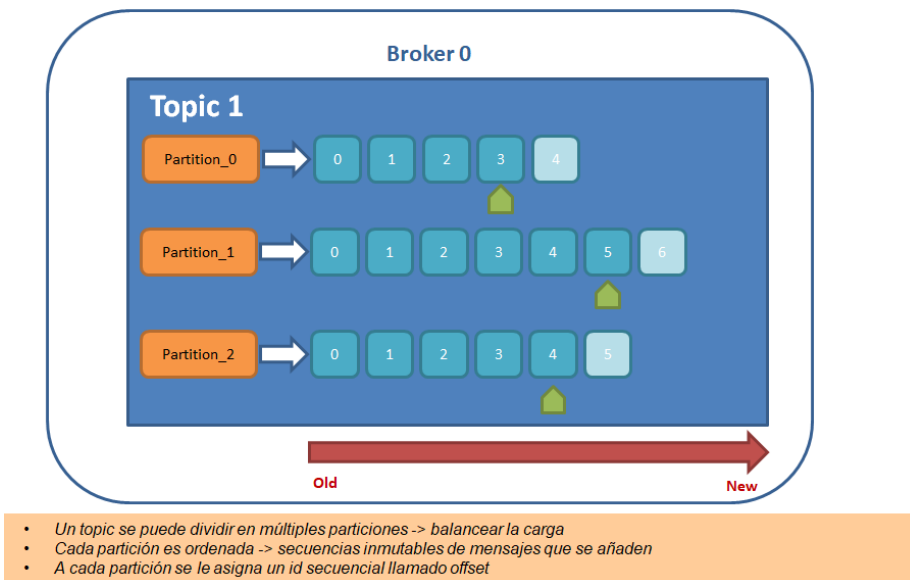
- Cada record tiene un periodo de retención
- Tiene diferentes "estados" :
 - **CONFIRMADO** : cuando todas las particiones ISR (In-Sync Replica) tienen escrito el mensaje en su log
 - **NO CONFIRMADO** : cuando al menos una partición ISR no tiene escrito el mensaje en su log

Diagrama Conceptual "Stream de Mensajes"



TOPIC

Diagrama Conceptual "Un único tema/topic con 3 particiones en un broker"



- Un único nodo / broker
- Nombre de la partición : "Topic 1"
- Número de particiones : 3

ZOOKEEPER

Software que proporciona un servicio de coordinación de alto rendimiento para aplicaciones distribuidas

- Manager/gestor del clúster (coordina la topología de los brokers / clúster)

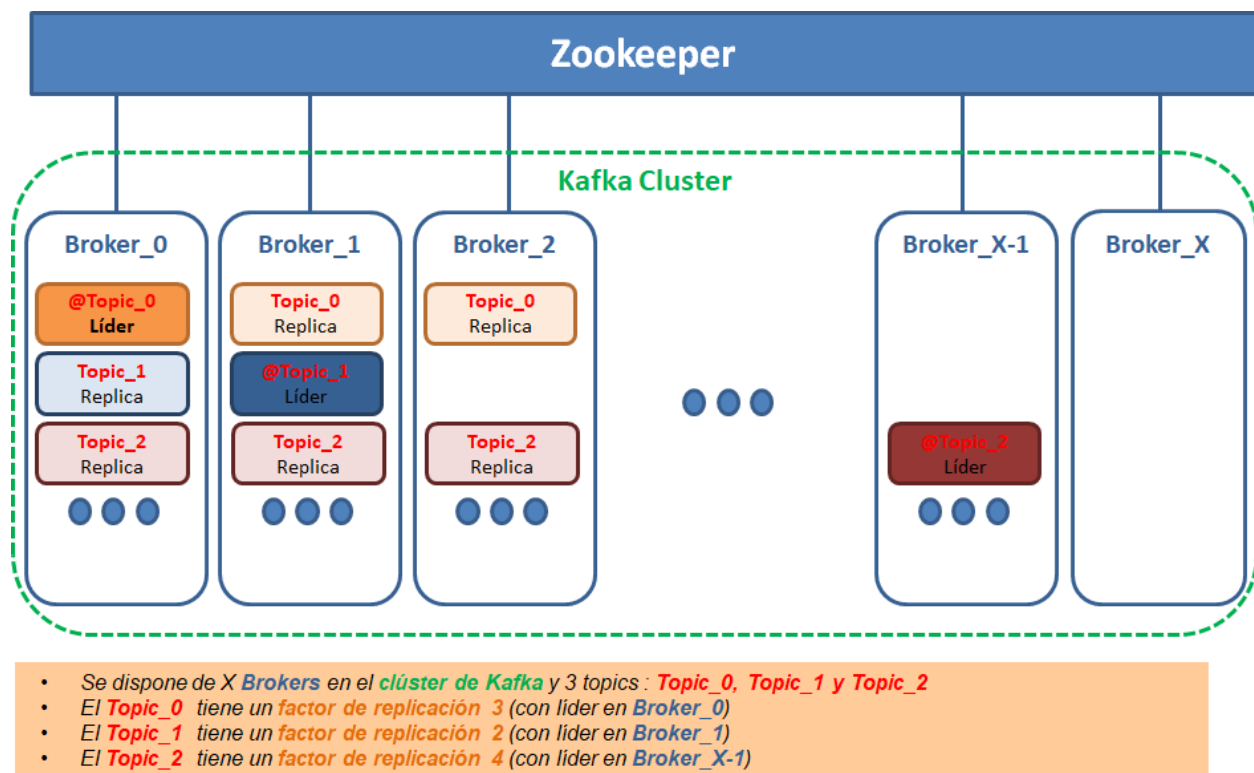
- Almacén Clave-Valor distribuido con los aspectos de control de la plataforma

Proporciona un servicio centralizado para la gestión de la configuración, registro de cambios, servicio de descubrimiento, etc. (se entera de la incorporación de un broker, cuando muere un broker, cuando se incorpora un topic, estado de salud de las particiones, etc.)

- Requiere ser el primer elemento en arrancar si se requiere una coordinación distribuida
- Intercambia metadatos con : brokers, productores y consumidores
 - *Direcciones de brokers*
 - *Offsets de los mensajes consumidos*
 - *Descubrimiento y control de los brokers del clúster*
 - *Detección de la carga de trabajo y se produce asignación por cercanía o bien por tener una menor ocupación*
 - *Etc.*
- Proporciona una vista sincronizada de la configuración del clúster

Ejemplo de configuración en kafka

Diagrama Conceptual "3 temas/topics con diferentes factores de replicación en X brokers"



- X nodos / brokers dentro de un clúster

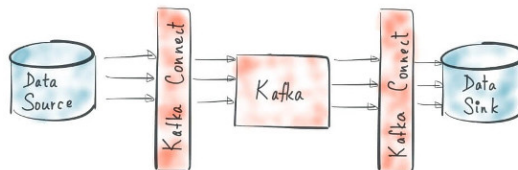
- Número de topics: 3
- Cada topic sólo tiene un partición
- Cada partición tiene diferentes factores de replicación
- Existen particiones leader y particiones follower o réplica
- Cada partición tiene un partición leader en broker diferentes aunque podían haber coincidido en el mismo

Alternativas de servicios de mensajería

RabbitMQ, Spring Cloud Stream, Kafka Streams, Apache Kafka, Amazon Kinesis, Google PubSub, Azure Event Hubs, Apache RocketMQ.

Ecosistema de Apache Kafka

- Kafka Core
 - Broker
 - Producer API, Consumer API, Admin API
 - Herramientas de administración
- Kafka Conncet



- Kafka Streams API
- Mirror Maker
- REST Proxy para HTTP y Kafka
- Kafka Security

Servicios de Cloud con soporte para Apache Kafka

Confluent Cloud: Fully Managed Kafka as a Cloud-Native Service


Set data in motion while avoiding the headaches of infrastructure management. Focus on what matters: your business. Confluent Cloud is a fully managed, cloud-native Kafka service for connecting and processing all of your data, everywhere it's needed.

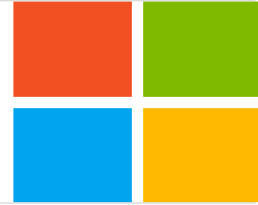
🔗 https://www.confluent.io/confluent-cloud/?utm_medium=sem&utm_source=google&utm_campaign=ch.sem_br.nonbrand_tp.prs_tgt.kafka_mt.xct_rgn.latam_lng.eng_dv.all_con.kafka-cloud&utm_term=kafka%20cloud&creative=&device=c&placement=&gclid=CjwKCAiAI-6PBhBCEiwAc2GOVIVZ1qkeoS_N_-3nXc7VV0r3FHgZHdKhCQ_Uv__c1P-y5ZOvAZHRKxoCVggQAvD_BwE



Información general de Apache Kafka en Confluent Cloud: soluciones de partners de Azure

Apache Kafka para Confluent Cloud es una oferta de Azure Marketplace que proporciona Apache Kafka como servicio. Está totalmente administrado para que pueda centrarse en la creación de aplicaciones en lugar de en la administración de los clústeres.

 <https://docs.microsoft.com/es-es/azure/partner-solutions/apache-kafka-confluent-cloud/overview>



CloudKafka - Apache Kafka Message streaming as a Service

Top 3 CloudKafka features CloudKafka automates every part of setup, running and scaling of Apache Kafka. Just click a button and you'll have a fully managed Kafka cluster up and running within two minutes. Our control panel offers various tools and integrations

 <https://www.cloudkafka.com/>

 cloudkafka

Message streaming as a Service

Powered by Apache Kafka, fully managed,
epic performance & superior support


Uso de Streaming con Apache Kafka

Oracle Cloud Infrastructure Streaming permite a los usuarios de Apache Kafka descargar la configuración, el mantenimiento y la gestión de infraestructura que su propio cluster de Zookeeper y Kafka requiere. Streaming es compatible con la mayoría de las API de Kafka, lo que permite utilizar aplicaciones escritas para Kafka para enviar mensajes y recibir mensajes del servicio Streaming sin tener que reescribir el

 <https://docs.oracle.com/es-ww/iaas/Content/Streaming/Tasks/kafkacompatibility.htm>

odooClusters

We use cookies to help optimize the website and give you the best experience. Privacy Policy Fully & Proactively Managed 7-Day Free Trial 24/7 Technical Support Starting at \$4.99/m Deploy in Minutes

 https://www.kaclusters.com/?utm_source=google.com&utm_medium=ads&utm_term=apache%20kafka&gclid=CjwKCAiAl-6PBhBCEiwAc2GOVH8h1We7olxKf5axO2CnpquUQUYV6qHRk_8rMTIm2VJKIMZAskAxC1zsQAvD_BwE



AWS Marketplace: Apache Kafka on Confluent Cloud? - Pay As You Go

Focus on building apps and not managing Apache Kafka clusters with a scalable, resilient and secure event streaming platform. Event streaming with Kafka made simple on AWS.

 <https://aws.amazon.com/marketplace/pp/prodview-g5ujul6iovvcy>

 awsmarketplace

Recursos:

<https://cloud.google.com/learn/what-is-apache-kafka?cloudshell=false>

<https://www.youtube.com/watch?v=iM7aq8tWTmc>

<https://kafka.apache.org/documentation/#datacenters>

<https://enmilocalfunciona.io/aprendiendo-apache-kafka-parte-1/>

<https://enmilocalfunciona.io/aprendiendo-apache-kafka-parte-2-2/>

<https://enmilocalfunciona.io/aprendiendo-apache-kafka-parte-3-conceptos-basicos-extra/>

<https://enmilocalfunciona.io/aprendiendo-apache-kafka-parte-4/>

<https://spring.io/projects/spring-kafka#samples>

<https://docs.spring.io/spring-kafka/docs/current/reference/html/#getting-started>

<https://www.itdo.com/blog/kafka-para-gestionar-los-datos-en-tiempo-real-de-mis-servicios/#:~:text=El clúster Kafka almacena flujos,la persistencia de los datos.>

<https://enmilocalfunciona.io/aprendiendo-apache-kafka-parte-3-configuracion-con-replicacion/>