IT/Dev CONNECTIONS

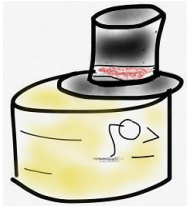
Using Kafka for Real-Time Data Ingestion with .NET

Kevin Feasel
Engineering Manager, Predictive Analytics
ChannelAdvisor

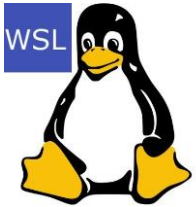
Delivered by KNect365 TMT an Informa business

Who Am I? What Am I Doing Here?

[Catallaxy Services]



[Curated SQL]

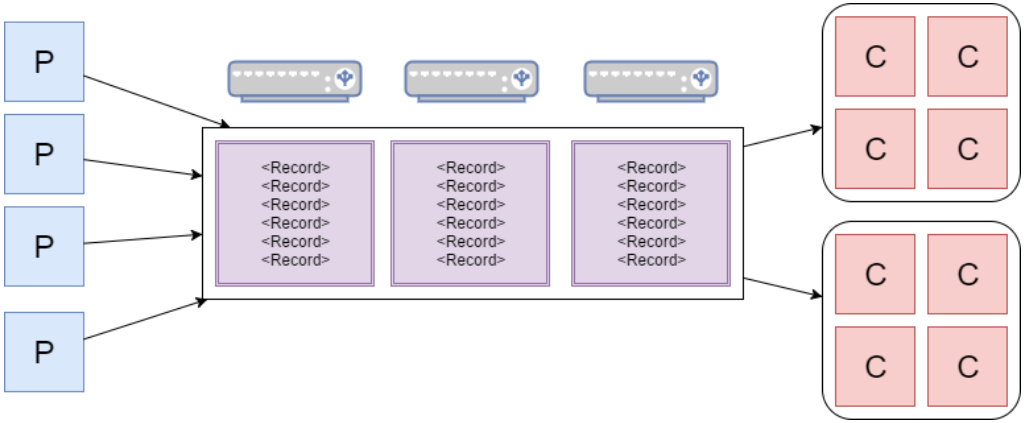


[We Speak Linux]

[@feaselkl]

Apache Kafka

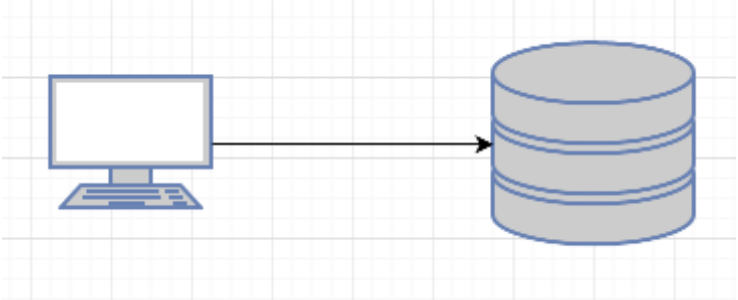
Apache Kafka is a message broker on the Hadoop stack. It receives messages from producers and sends messages to consumers.



Everything in Kafka is distributed.

Why Use A Broker?

Suppose we have two applications which want to communicate. We connect them directly.

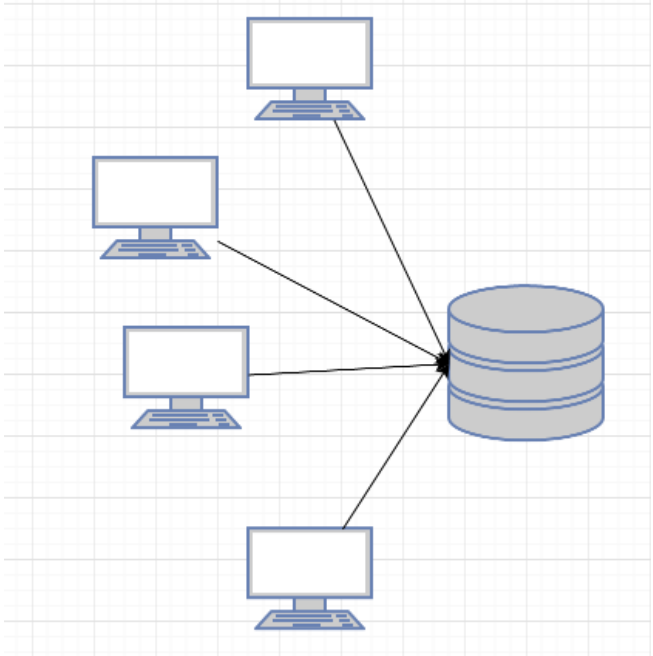


Works great at low scale--it's easy to understand, easy to work with, and has fewer working parts to break. But it hits scale limitations.

Why Use A Broker?

We then expand out.

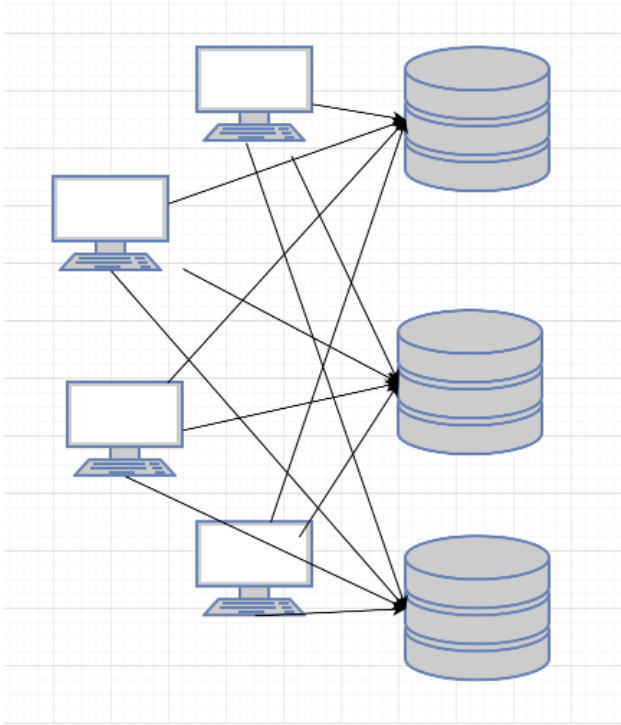
It is easy to expand this way as long as you don't overwhelm the DB; eventually you will.



Why Use A Broker?

We then expand out. Again.
It takes some effort here: we need to manage connection strings and write to the correct DB.

But it's doable and expands indefinitely.

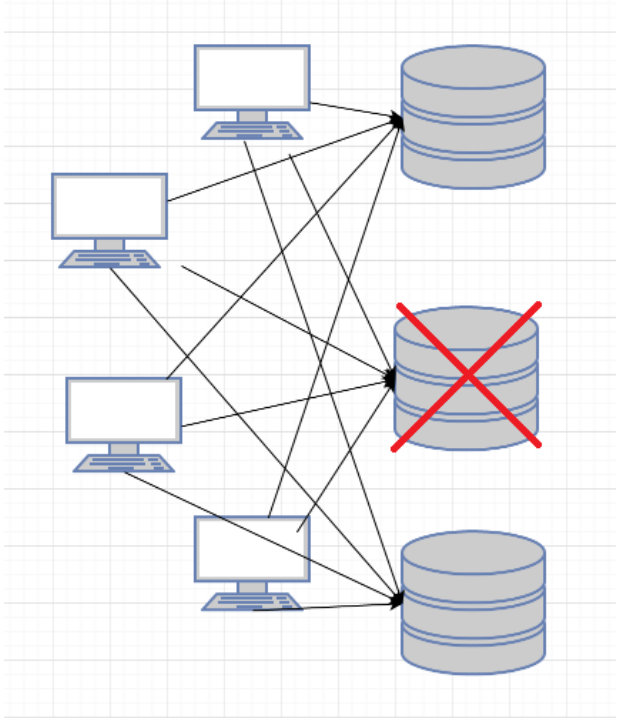


Why Use A Broker?

But what happens when a consumer (database) goes down for too long?

- Producer drops messages
- Producer holds messages (until it runs out of disk)
- Producer returns error

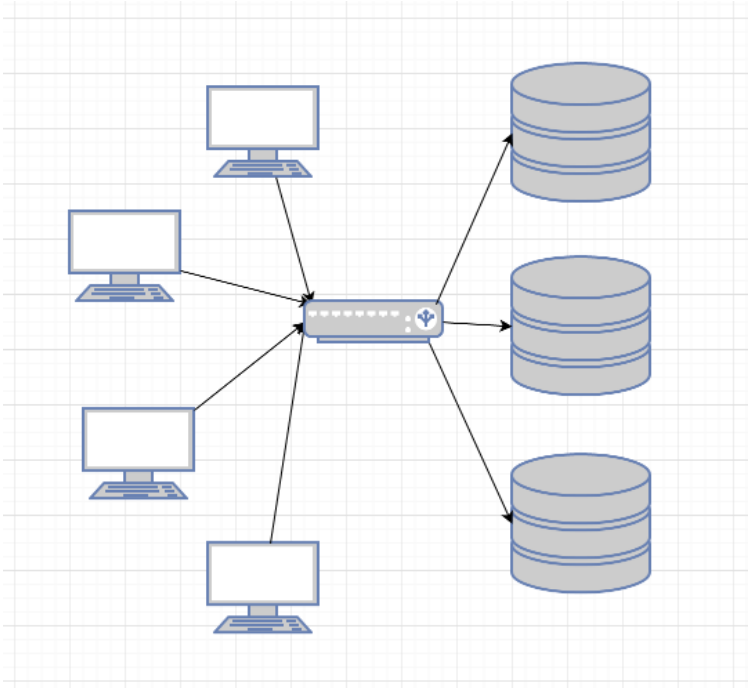
There's a better way.



Why Use A Broker?

Brokers take messages from producers and feed messages to consumers.

Brokers deal with the jumble of connections, let us be resilient to producer and consumer failures, and help with scale-out.



Motivation

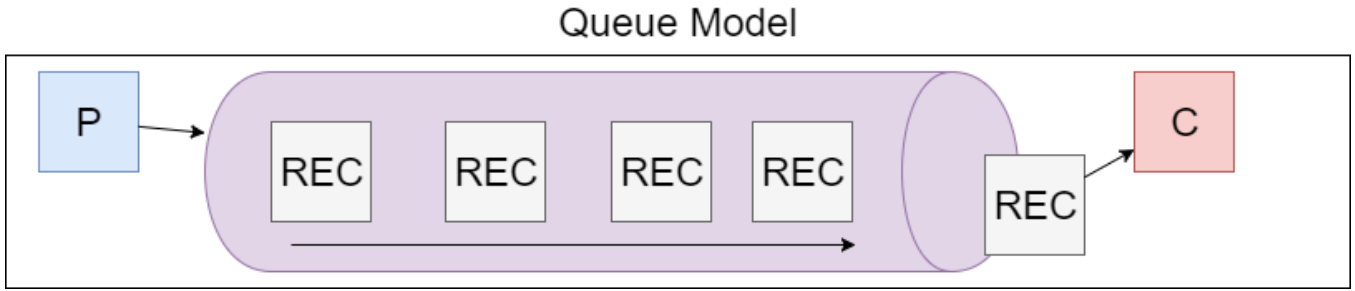
Today's talk will focus on using Kafka to ingest, enrich, and consume data. We will build .NET applications in Windows to talk to a Kafka cluster on Linux.

Our data source is flight data. I'd like to ask a few questions, with answers split out by destination state:

1. How many flights did we have in 2008?
2. How many flights' arrivals were delayed?
3. How many minutes of arrival delay did we have?
4. Given a flight with a delay, how long can we expect it to be?

Kafka Concepts

Most message brokers act as queues.

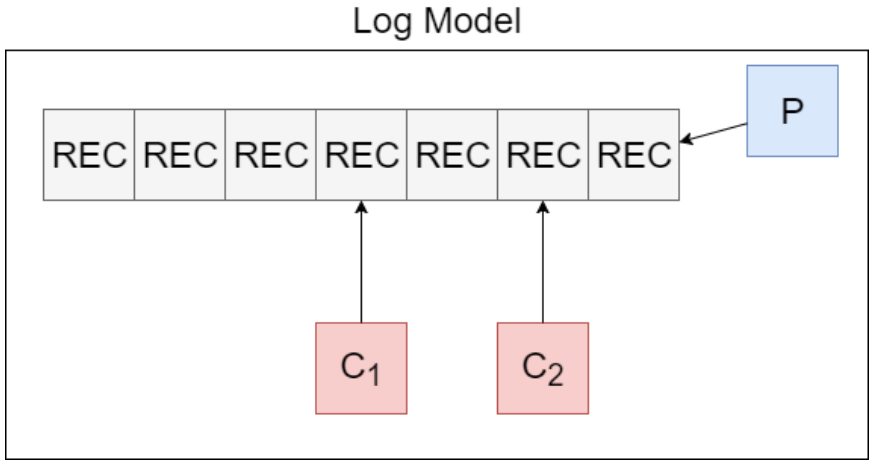


Kafka Concepts

Kafka is a log, not a queue.

Multiple consumers may read the same message and a consumer may re-read messages.

Think microservices and replaying data.



Kafka Concepts

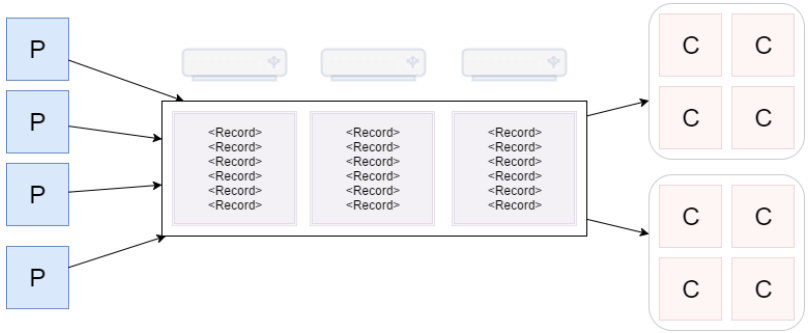
Brokers foster communication between producers and consumers. They store the produced messages and keep track of what consumers have read.

Kafka Concepts

Topics are categories or feeds to which messages get published. Topics are broken up into partitions. **Partitions** are ordered, immutable sequences of records.

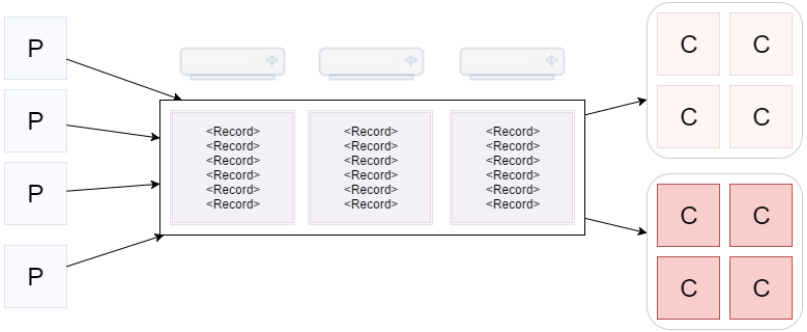
Kafka Concepts

Producers push messages to Kafka.



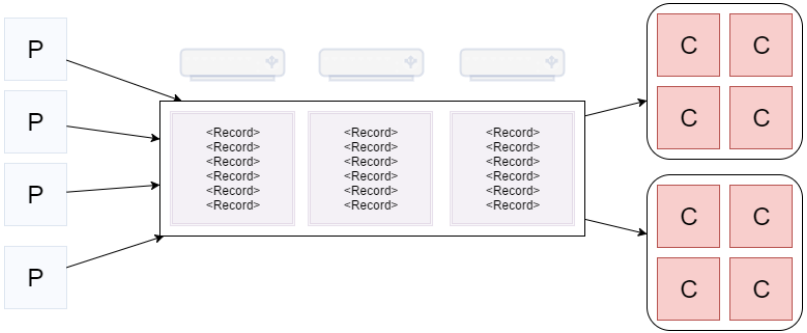
Kafka Concepts

Consumers read messages from topics.



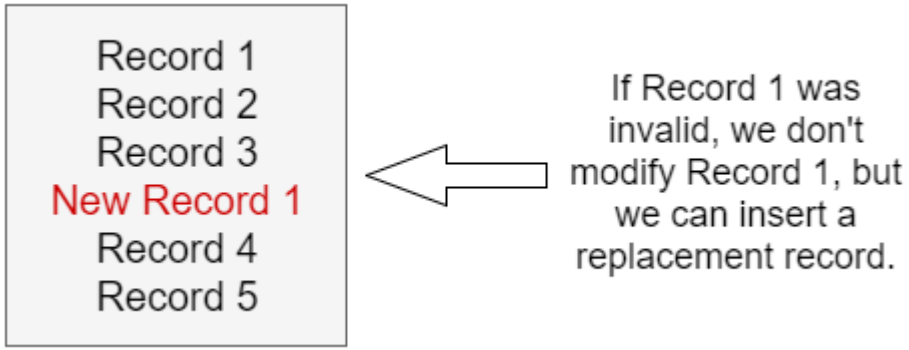
Kafka Concepts

Consumers enlist in consumer groups. **Consumer groups** act as "logical subscribers" and Kafka distributes load to consumers in a group.



Kafka Concepts

Records in partitions are **immutable**. You do not modify the data, but can add new rows.



Kafka Concepts

- Consumers should know where they left off. Kafka assists by storing consumer group-specific last-read pointer values per topic and partition.
- Kafka retains messages for a certain (configurable) amount of time, after which point they drop off.
- Kafka can also garbage collect messages if you reach a certain (configurable) amount of disk space.

The Competition

- MSMQ and Service Broker: queues in Microsoftland
- Amazon Kinesis and Azure Event Hub: Kafka as a Service
- RabbitMQ: complex routing & guaranteed reliability
- Celery: distributed queue built for Python
- ZeroMQ: socket-based distributed queueing
- Queues.io lists dozens of queues and brokers

Building A Producer

Our first application reads data from a CSV and pushes messages onto a topic.

This application will not try to understand the messages; it simply takes data and pushes it to a topic.

Building A Producer

I chose Confluent's Kafka .NET library (nee RDKafka-dotnet) as my library of choice.

There are several libraries available, each with their own benefits and drawbacks. This library serves up messages in an event-based model and has official support from Confluent, so use this one.

Demo Time

Building An Enricher

Our second application reads data from one topic and pushes messages onto a different topic.

This application provides structure to our data and will be the largest application.

Building An Enricher

Enrichment opportunities:

1. Convert "NA" values to appropriate values: either a default value or None (not NULL!).
2. Perform lookups against airports given an airport code.
3. Converting the input CSV record into a structured type (similar to a class).
4. Outputting results as JSON for later consumers.

Demo Time

Building A Consumer

Our third application reads data from the enriched topic, aggregates, and periodically writes results to SQL Server.

We've already seen consumer code, so this is easy.

Demo Time

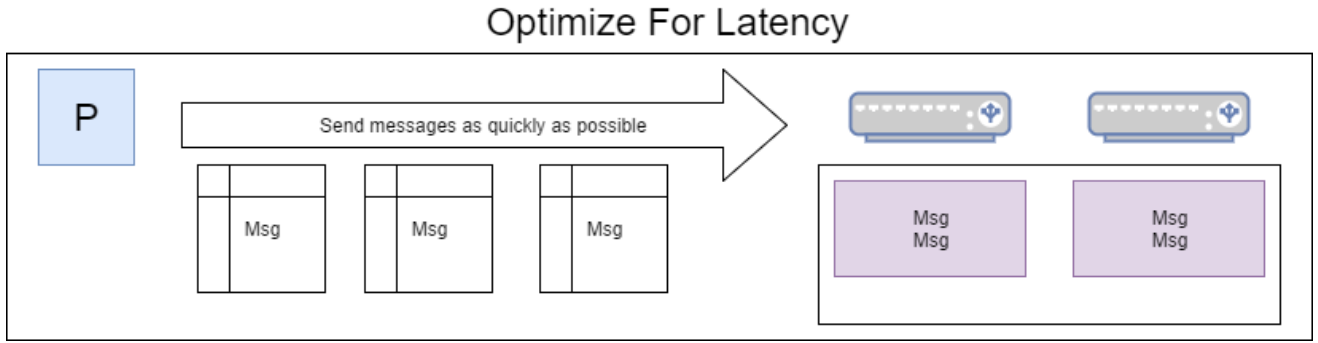
Kafka Performance

Basic tips:

- Maximize your network bandwidth! Your fibre channel will push a lot more messages than my travel router.
- Compress your data. Compression works best with high-throughput scenarios, so test first.
- Minimize message size. This reduces network cost.
- Buffer messages in your code using tools like `Collections.Concurrent.BlockingCollection`

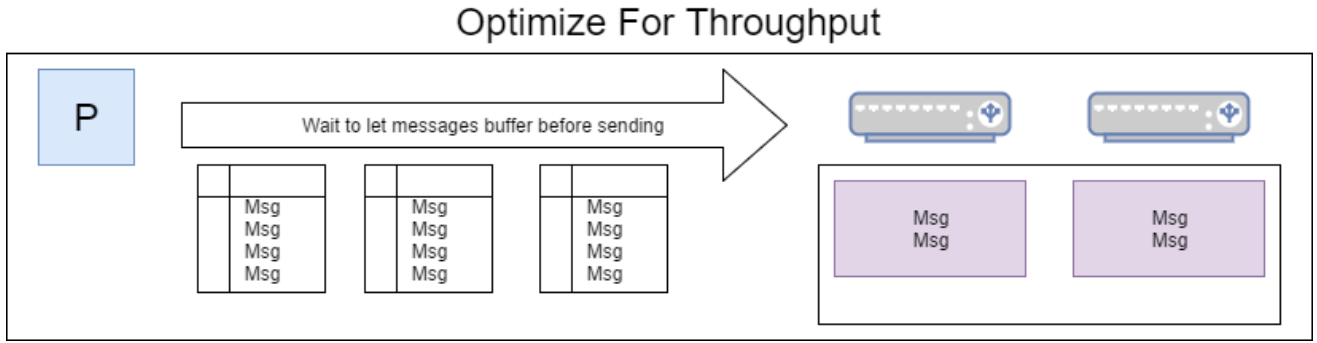
Throughput Versus Latency

Minimize latency when you want the most responsive consumers but don't need to maximize the number of messages flowing.



Throughput Versus Latency

Maximize throughput when you want to push as many messages as possible. This is better for bulk loading operations.



Throughput Versus Latency

Consumer config settings:

- `fetch.wait.max.ms`
- `fetch.min.bytes`

Producer config settings:

- `batch.num.messages`
- `queue.buffering.max.ms`

More, More, More

Kafka is a horizontally distributed system, so when in doubt, add more:

- More brokers will help accept messages from producers faster, especially if current brokers are experiencing high CPU or I/O.
- More consumers in a group will process messages more quickly.
- You must have at least as many partitions as consumers in a group! Otherwise, consumers may sit idle.

Wrapping Up

Apache Kafka is a powerful message broker. There is a small learning curve associated with Kafka, but this is a technology well worth learning.

To learn more, go here: https://CSmore.info/on/kafka

And for help, contact me:

feasel@catallaxyservices.com | @feaselkl