KubeCon | CloudNativeCon

North America 2024

Using OpenTelemetry for Deep Observability Within Messaging Queues

Shivanshu Raj Shrivastava, SigNoz
Ekansh Gupta, SigNoz

KubeCon CloudNativeCon
North America 2024

About the speakers

Ekansh Gupta

Software Engineer, SigNoz (YC W21)

Shivanshu Raj Shrivastava

Founding Engineer SigNoz (YC W21)
CNCF ambasador
Member & contributor opentelemetry

Talk Outline

Talk Outline

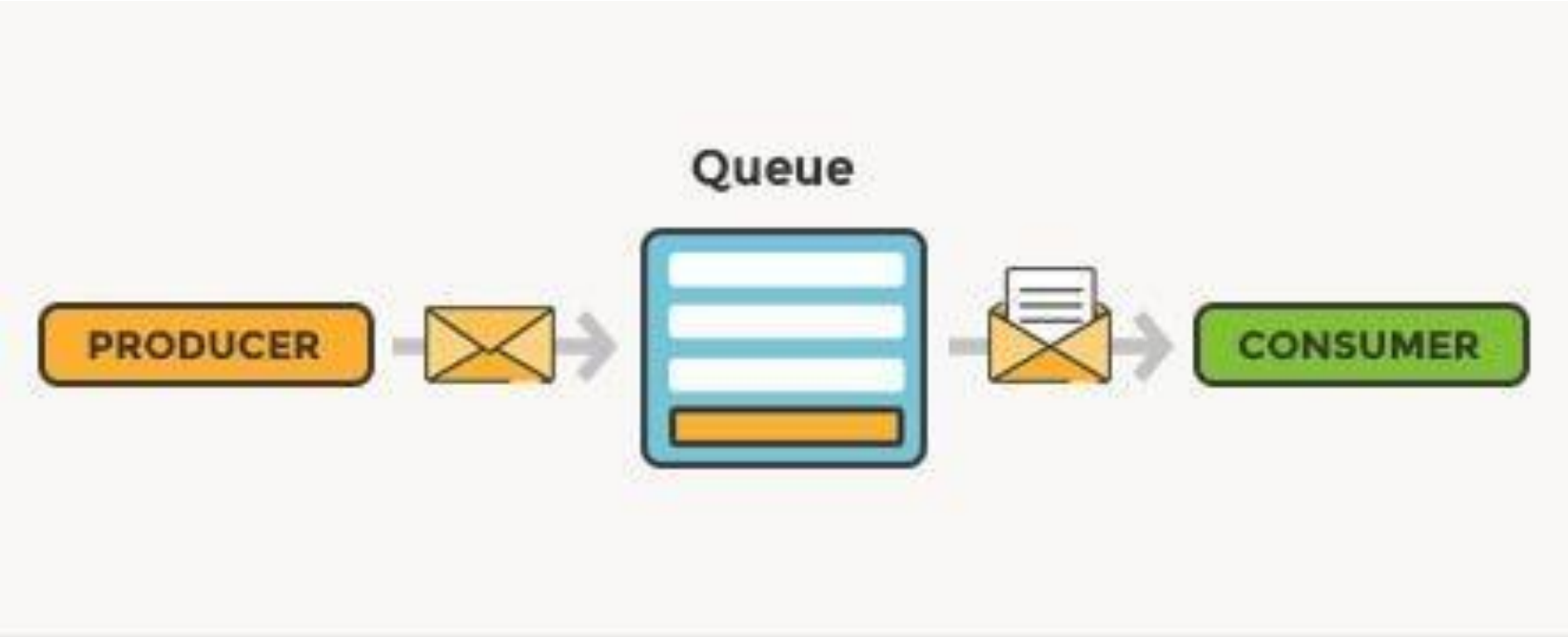
- What is Messaging Queue and where all we use it?
- What are the common problems that we face while using Messaging Queues such as Kafka / RabbitMQ
- Messaging queues are complex
- Lets deep dive into Kafka
- What we don’t see with Kafka Metrics
- The all powerful OTel
- Correlations and deeper insights into Messaging Queues.

What are Messaging Queues?

And

Where do we use it?

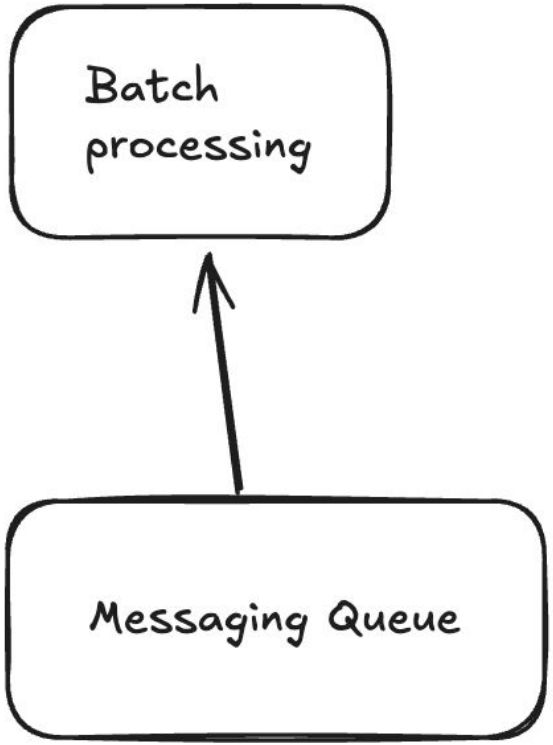
What are MQs and where do we use it?



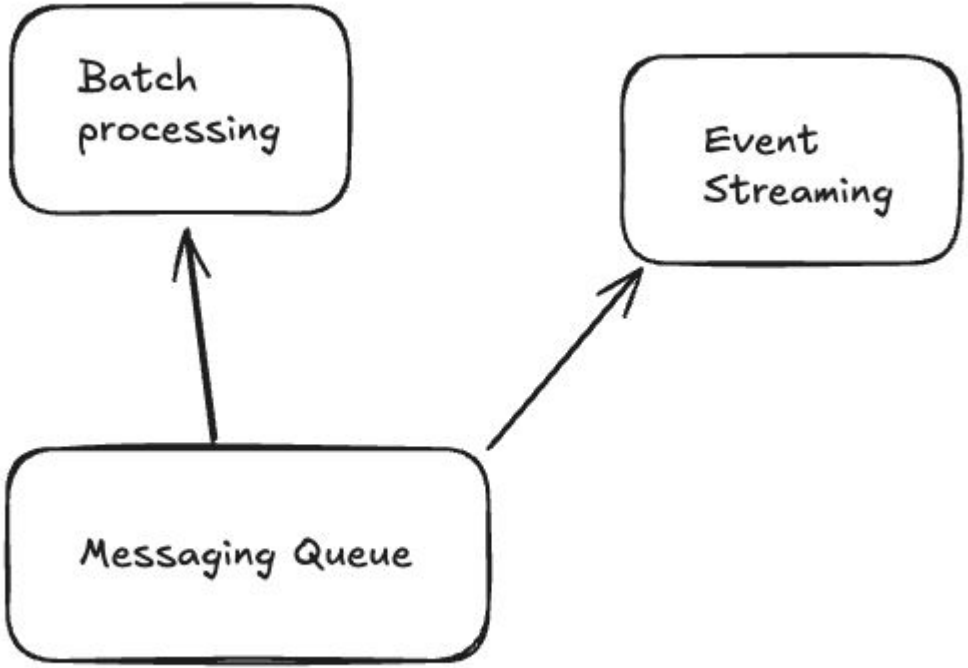
General Understanding of a Messaging Queue

Source: CloudAMQP

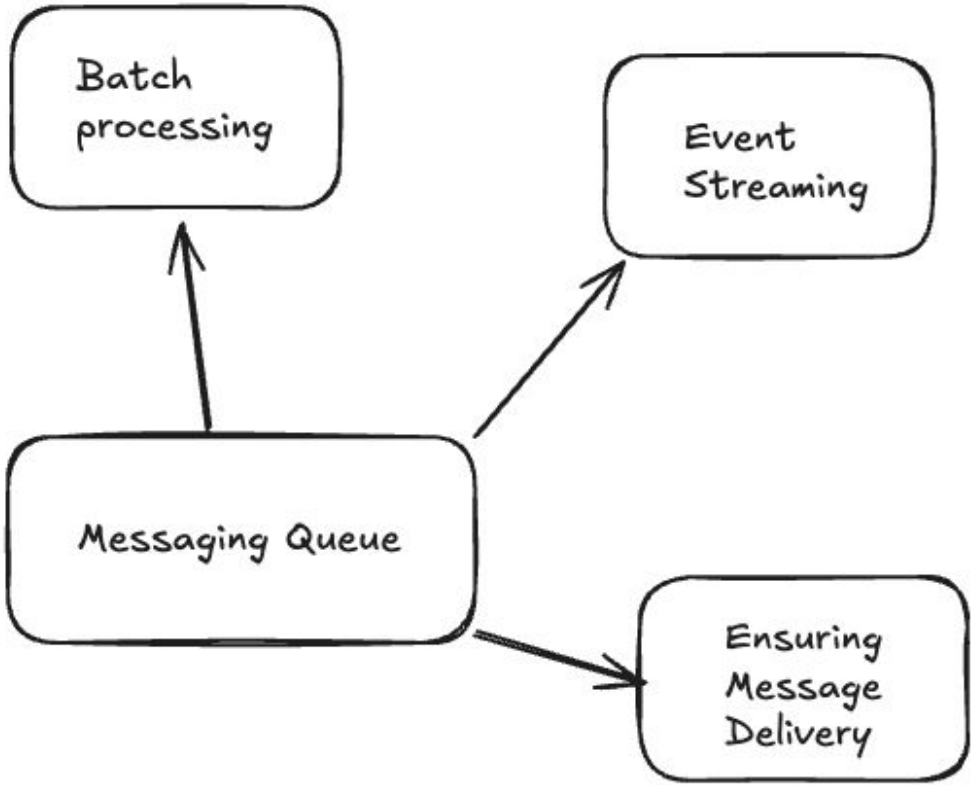
What are MQs and where do we use it?



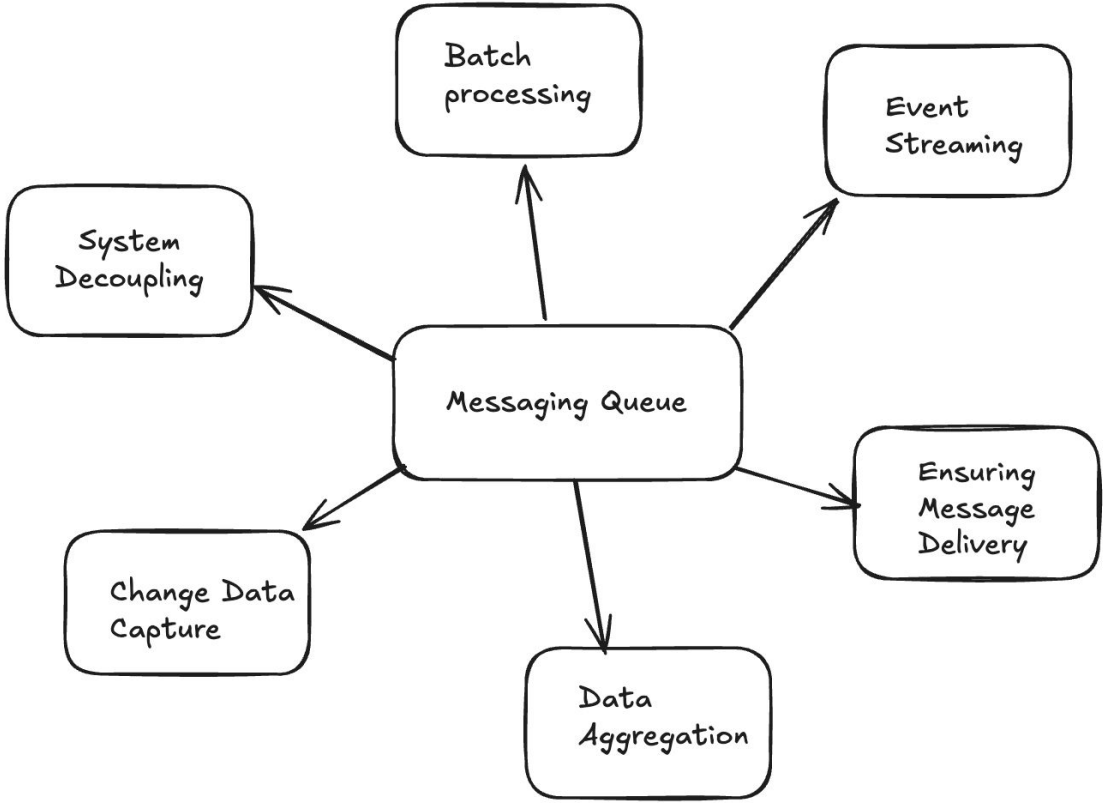
What are MQs and where do we use it?



What are MQs and where do we use it?



What are MQs and where do we use it?



Messaging Queues:

What can Actually go wrong?

What can go Wrong?

High Latency and Throughput Issues

No Visibility about Latency

Throttling Issues

Not Enough Metrics

Broker Failures

Consumer Lag and Back Pressure

Message Ordering Issues

Dead Letter Queues


ok ima fight
MESSAGING QUEUES
damn
KAFKA
GOT HANDS
imgflip.com

Messaging Queues are Complex

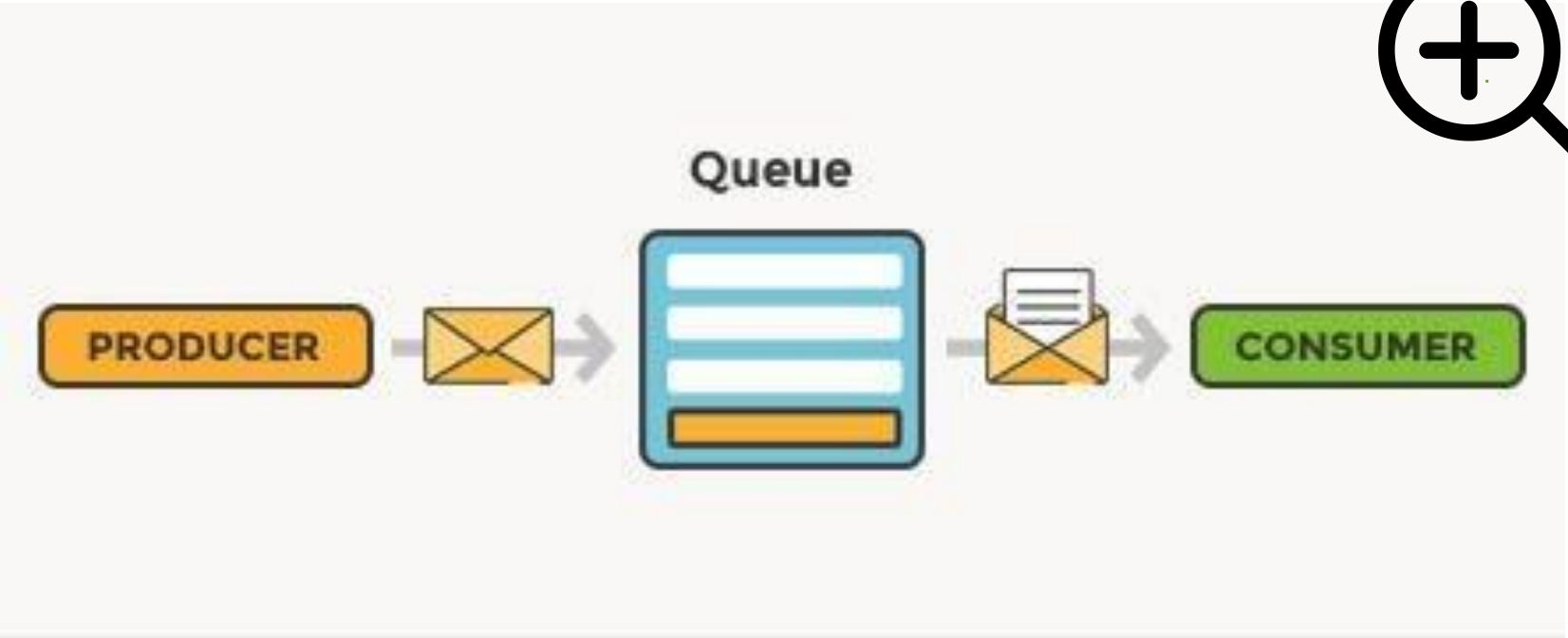
Messaging Queues are Complex!

Developers often use **default** configurations, while fine tuning with **deeper observability** can help them **scale better**

Knowing **when** and **how** things are breaking, can **help in fixing this faster** :)

Let’s Dive deeper into Messaging Queue

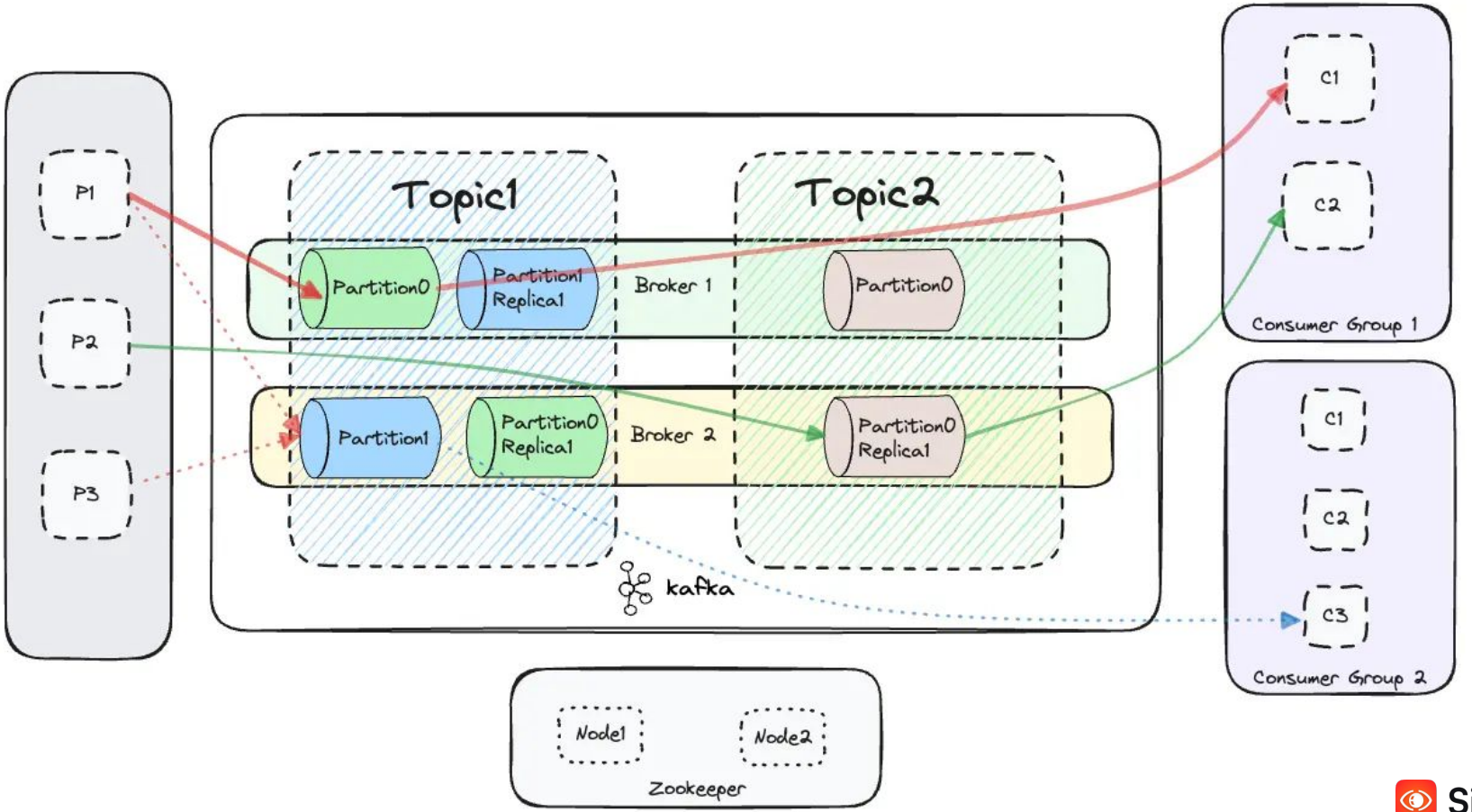
Messaging Queue



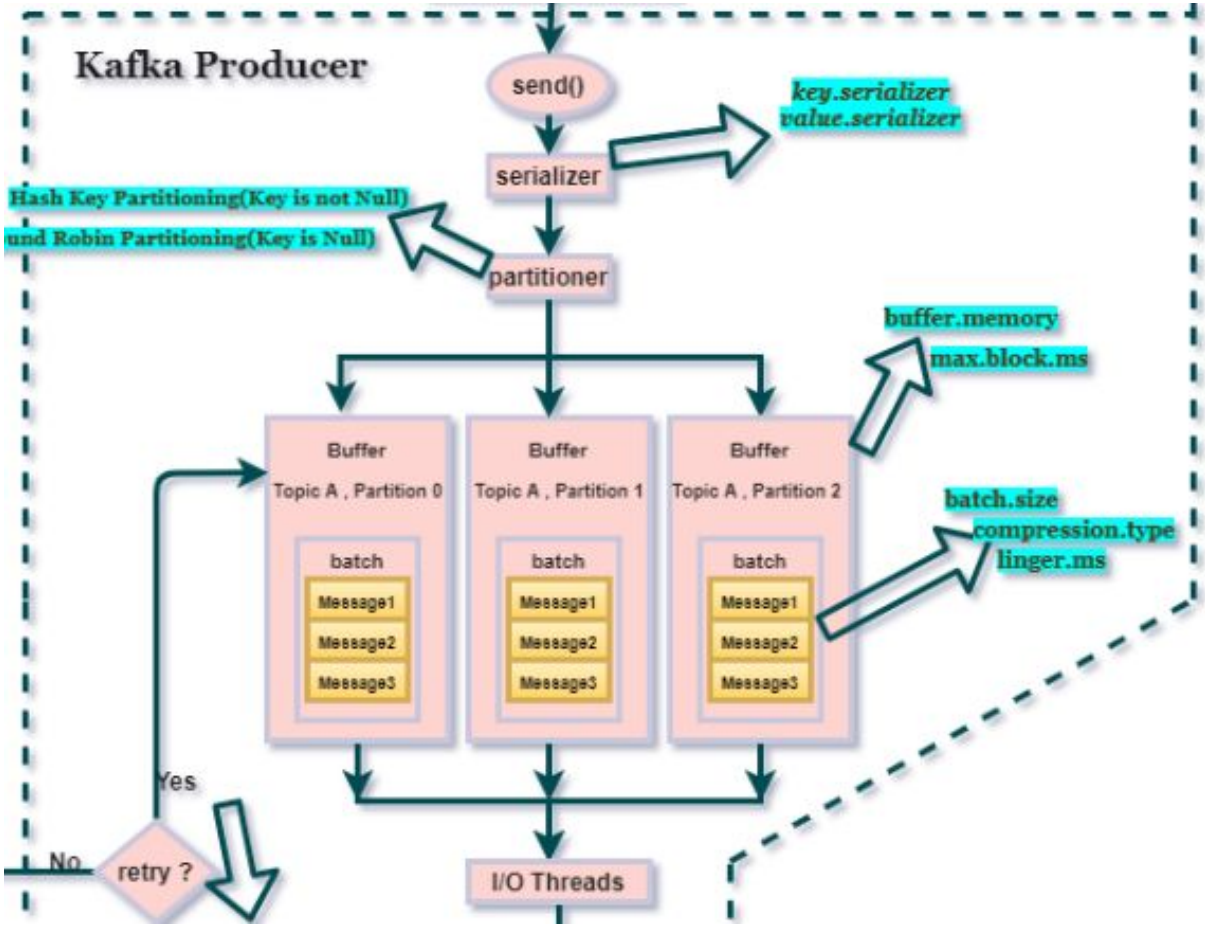
General Understanding of a Messaging Queue

Source: CloudAMQP

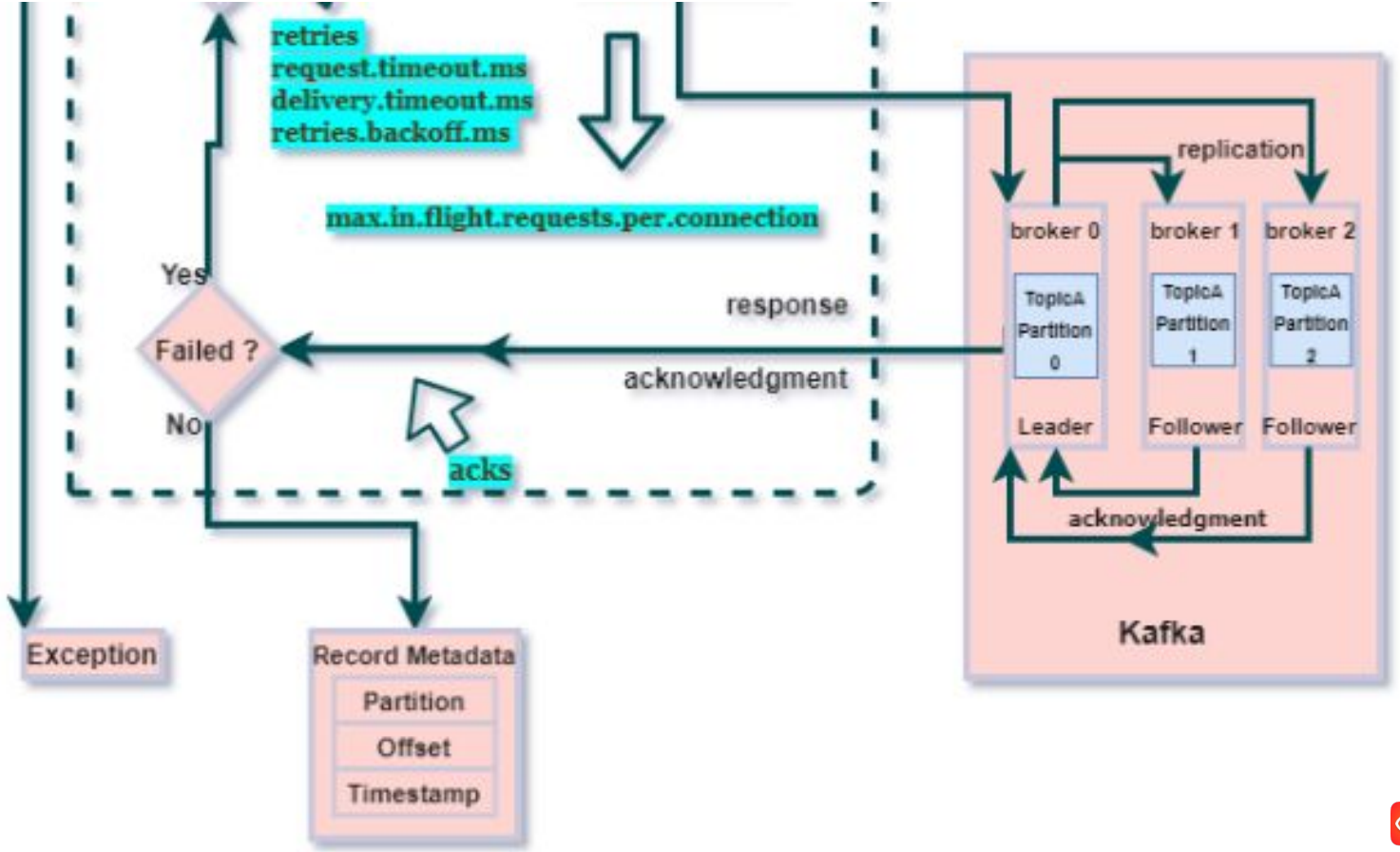
Kafka



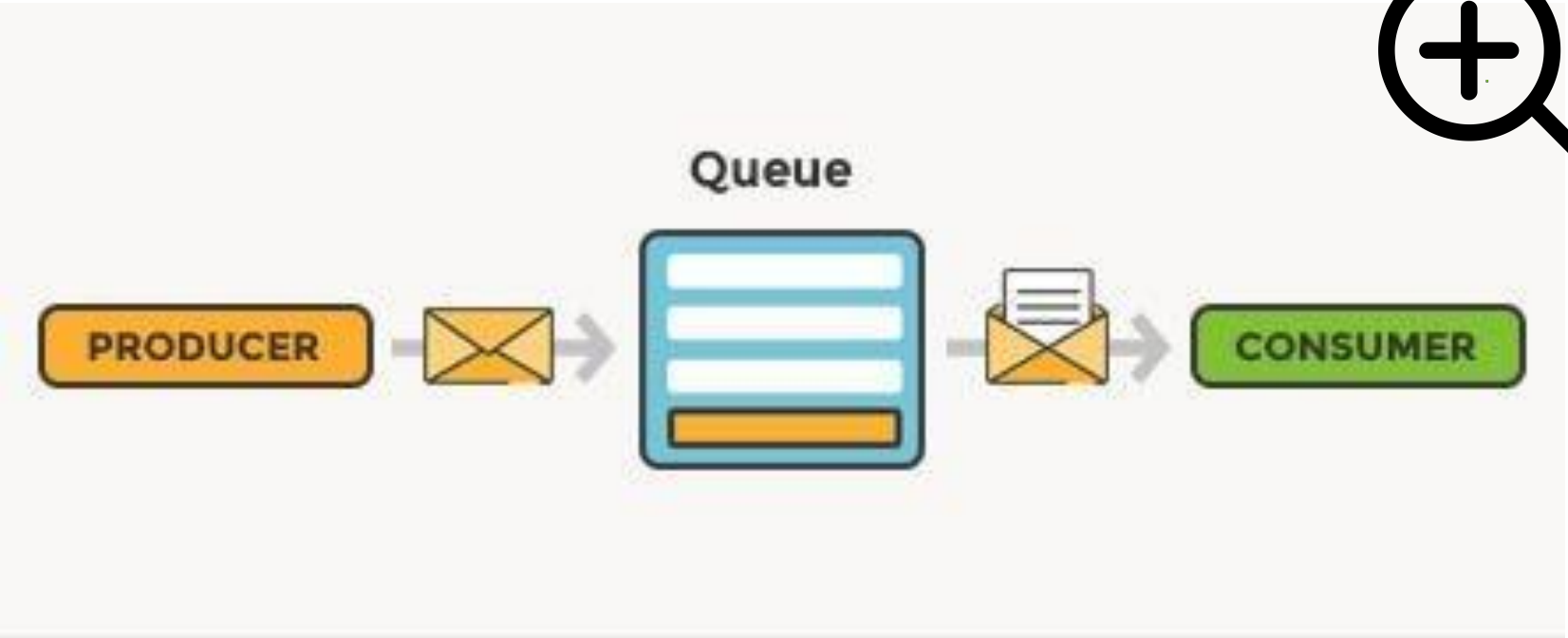
Kafka



Kafka


retries
request.timeout.ms
delivery.timeout.ms
retries.backoff.ms
max.in.flight.requests.per.connection
Yes
Failed ?
No
response
acknowledgment
acks
Exception
Record Metadata
Partition
Offset
Timestamp
replication
broker 0
broker 1
broker 2
TopicA Partition 0
TopicA Partition 1
TopicA Partition 2
Leader
Follower
Follower
acknowledgment
Kafka

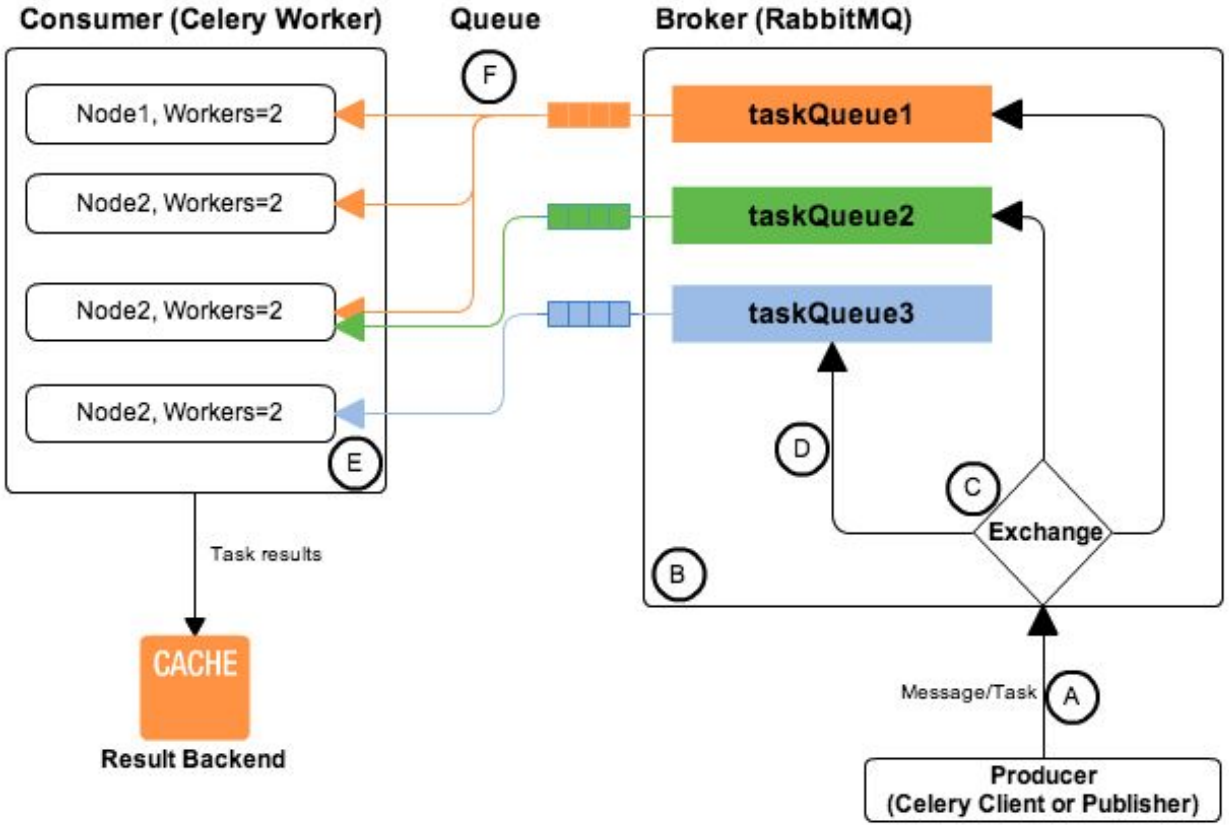
Messaging Queue



General Understanding of a Messaging Queue

Source: CloudAMQP

RabbitMQ + Celery



So what we don’t see with Broker Metrics?

(Un)Available Metrics

- **Kafka does not directly expose producer-side latency metrics per partition.**
  - No one ever knows why there is high producer latency.
  - There is no visibility if there's a problem with Kafka itself or producer.
  - Difficulty tracking producer throughput across different topics
  - Complex to monitor batch sizes and compression ratios
  - Complex to find the correlation between different partitions

(Un)Available Metrics

- **End-to-End Latency (How much time it took for producer to send a message and a consumer to acknowledge it)**
  - Difficult to track processing time per message
  - No direct correlation between consumer performance and resource utilization
  - Complex to correlate broker metrics with client-side issues

(Un)Available Metrics

- **Consumer Group Lag correlation with spans**
  - No visibility of which partition is causing the lag?
  - Is the lag caused by consumer application, network problems, or imbalances between producer and consumer performance.
  - Manual implementation needed for trace context propagation.
  - Limited visibility into consumer group rebalancing events

(Un)Available Metrics

- **Infrastructure Monitoring Gaps**
  - Complex relationship between JVM metrics and Kafka performance
  - Difficult to correlate network issues with message delivery problems
  - Limited visibility into disk I/O patterns
  - Difficult to track partition growth patterns

(Un)Available Metrics

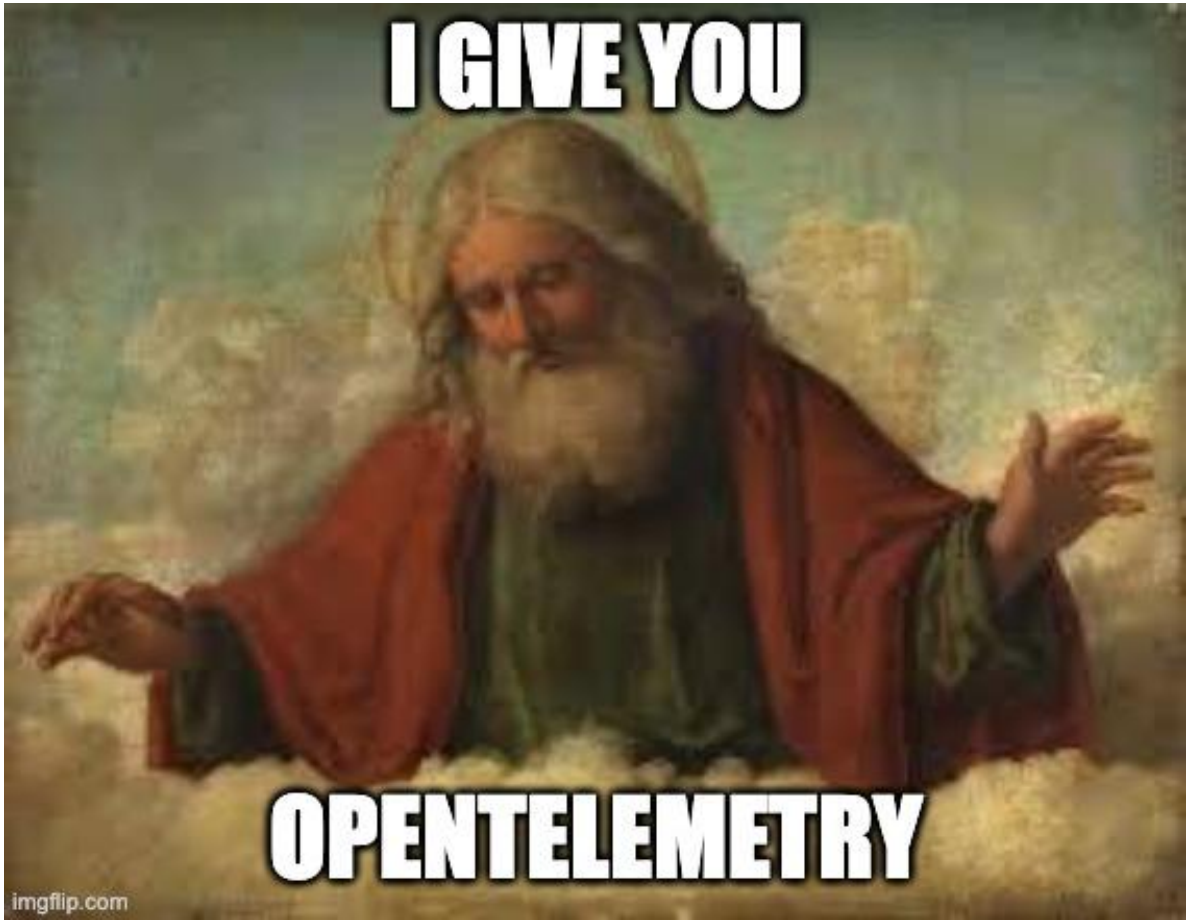
Metrics -> Trace

Traces -> Logs

Traces -> Metrics

All these correlations are hard
to achieve natively in messaging queues

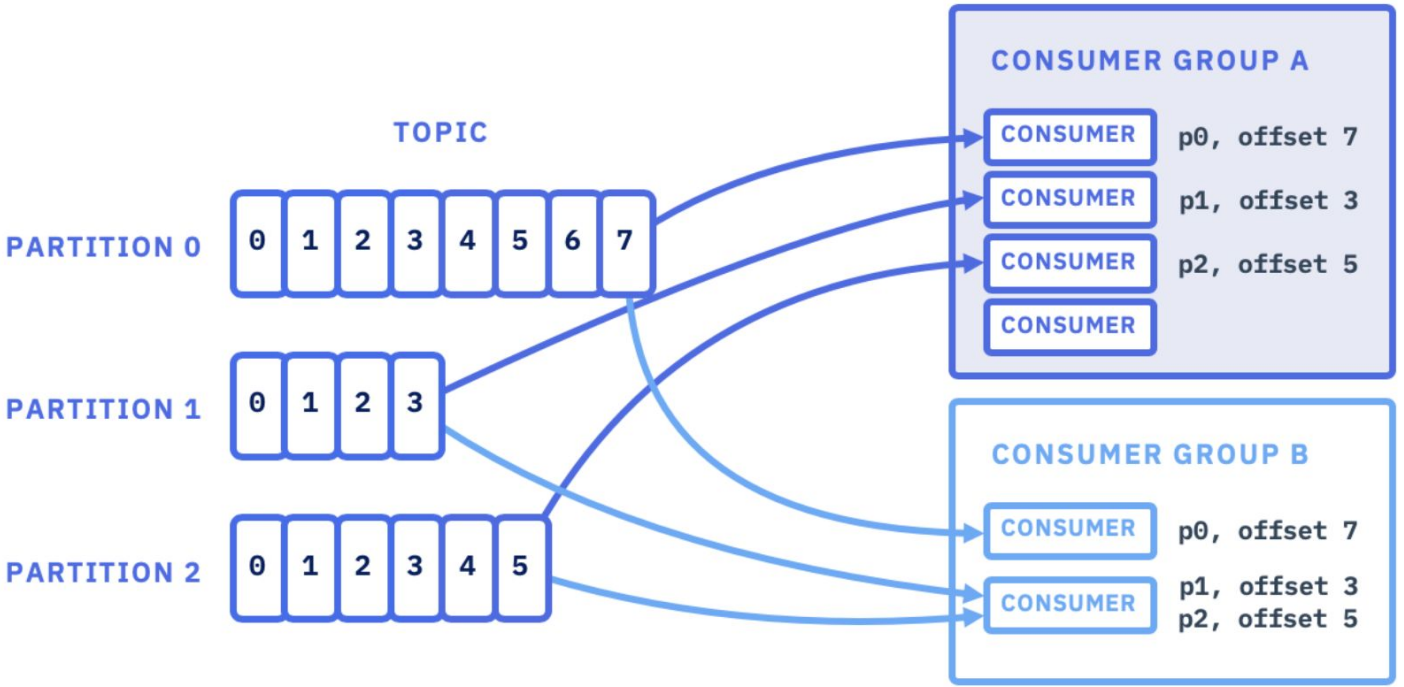
What is the answer to this Problem?


I GIVE YOU
OPENTELEMETRY
imgflip.com

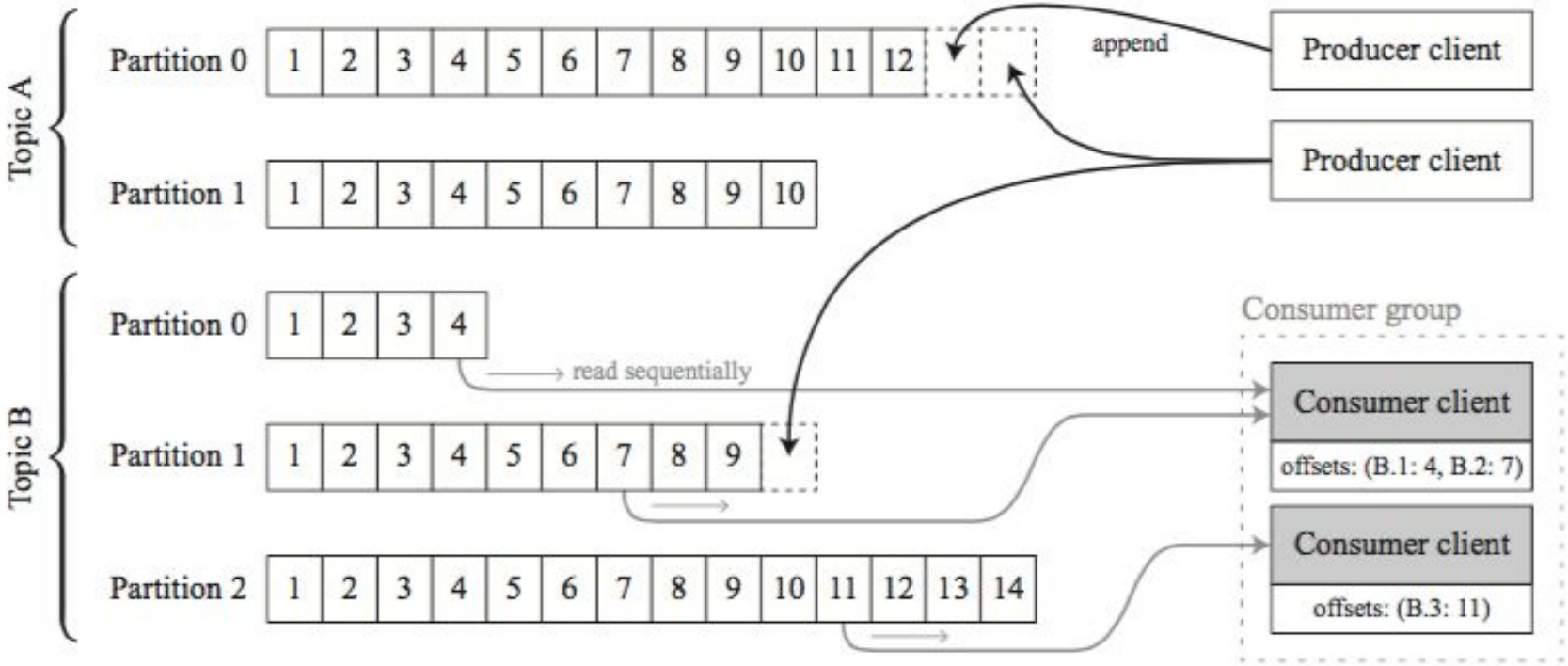
Common problems

- **Consumer Group Lag correlation with spans**
  - No visibility of which partition is causing the lag?
  - Is the lag caused by consumer application, network problems, or imbalances between producer and consumer performance.
  - Manual implementation needed for trace context propagation.
  - Limited visibility into consumer group rebalancing events

Common problems

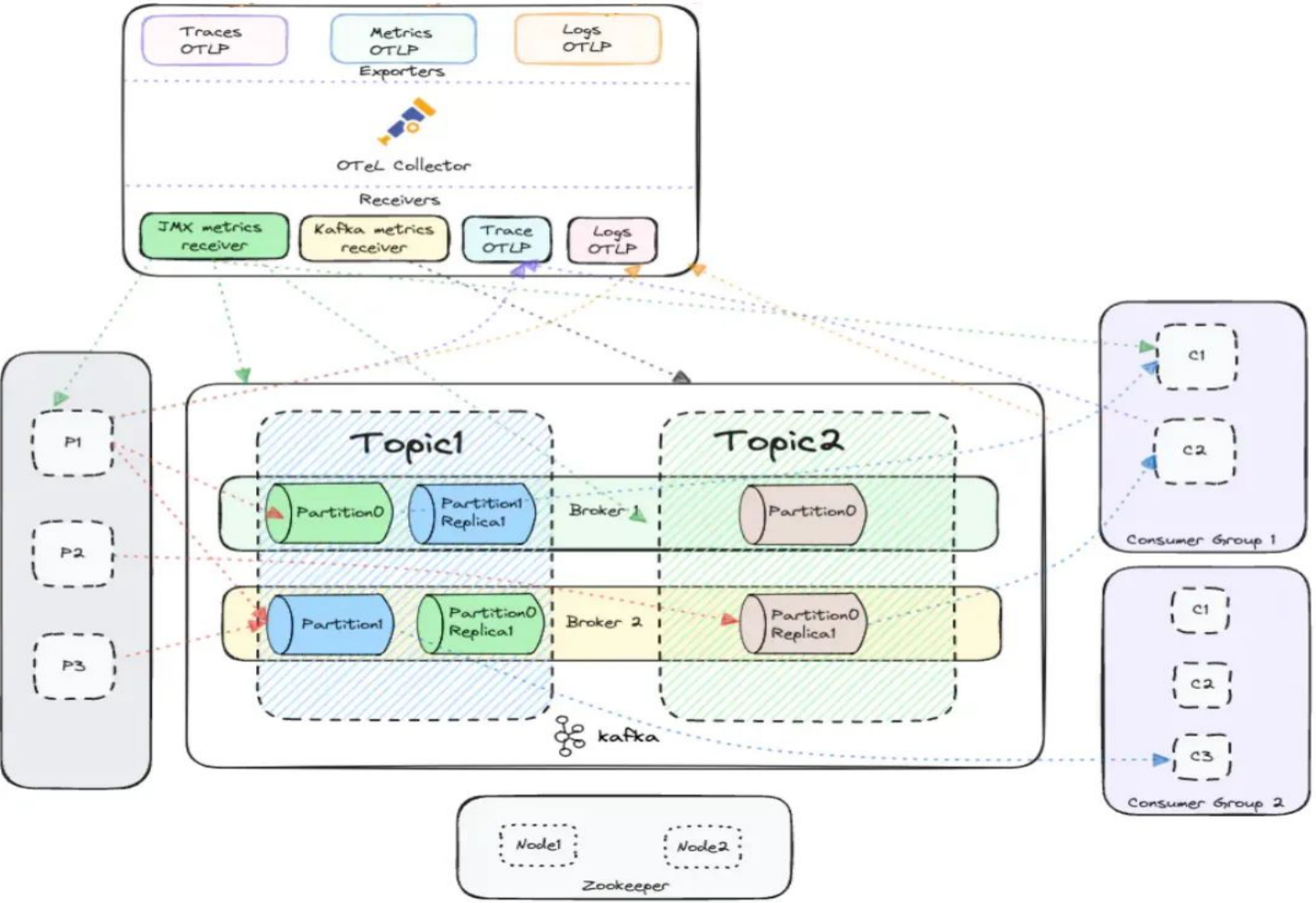


Noisy Neighbour


Topic A
Partition 0
1 2 3 4 5 6 7 8 9 10 11 12
Partition 1
1 2 3 4 5 6 7 8 9 10
append
Producer client
Producer client
Topic B
Partition 0
1 2 3 4
read sequentially
Partition 1
1 2 3 4 5 6 7 8 9
Partition 2
1 2 3 4 5 6 7 8 9 10 11 12 13 14
Consumer group
Consumer client
offsets: (B.1: 4, B.2: 7)
Consumer client
offsets: (B.3: 11)

Client-side OpenTelemetry

- **Stable API** with ability to export telemetry anywhere
- **Instrumentations** for popular clients and frameworks
- **Semantic conventions** to unify telemetry format


Traces OTLP
Metrics OTLP
Logs OTLP
Exporters
OTeL Collector
Receivers
JMX metrics receiver
Kafka metrics receiver
Trace OTLP
Logs OTLP
P1
P2
P3
Topic1
Topic2
Partition0
Partition1 Replica1
Broker 1
Partition0
Partition1
Partition0 Replica1
Broker 2
Partition0 Replica1
kafka
C1
C2
Consumer Group 1
C1
C2
C3
Consumer Group 2
Node1
Node2
Zookeeper

Kafka OTel Setup


```yaml
receivers:
  otlp:
    protocols:
      grpc:
        # OTeL receiver endpoint (grpc)
        endpoint: 127.0.0.1:4317
      http:
        # OTeL receiver endpoint (http)
        endpoint: 127.0.0.1:4318
  kafkametrics:
    brokers:
      # Kafka brokers endpoints
      - 127.0.0.1:9092
      - 127.0.0.1:9093
    protocol_version: 2.0.0
    scrapers:
      - brokers
      - topics
      - consumers
```

Kafka OTel Setup

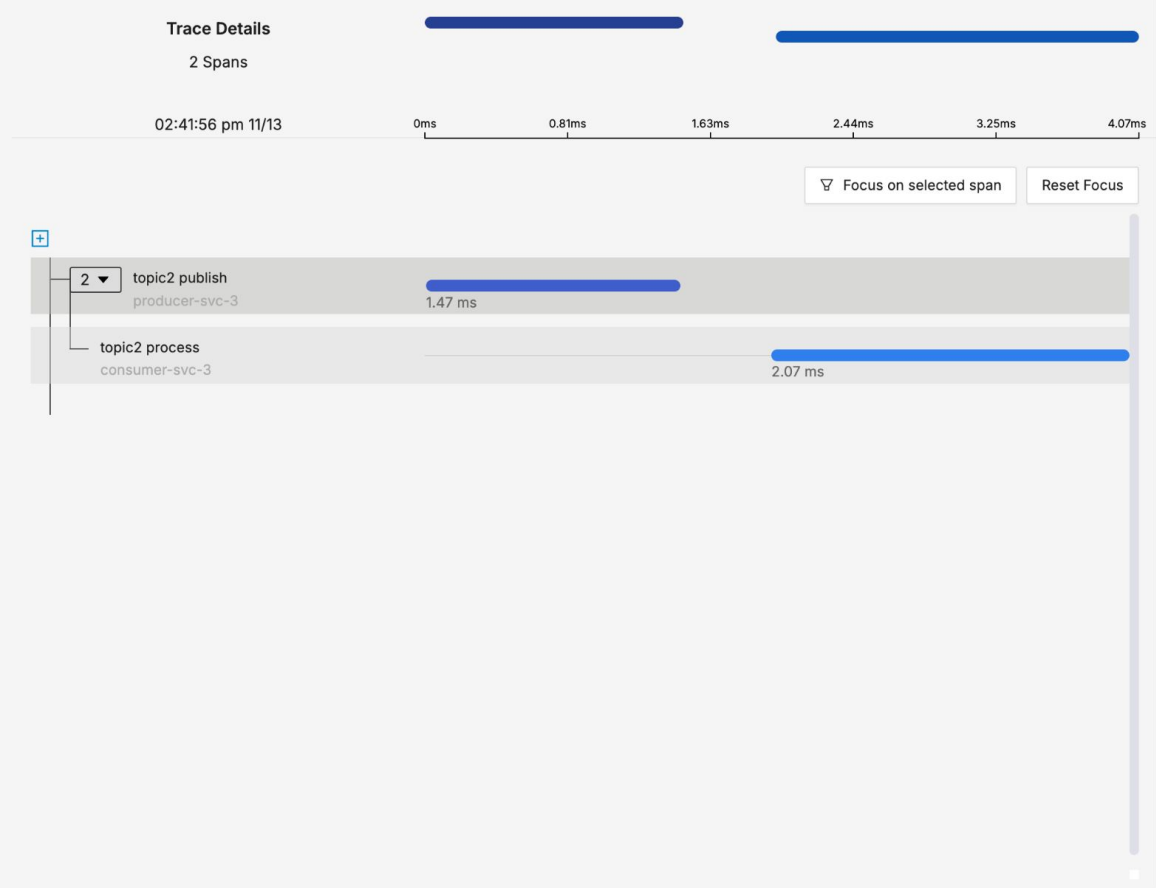

```
jmx:
  # configure the path where you installed opentelemetry-jmx-metrics receiver
  jar_path: ${PWD}/opentelemetry-jmx-metrics.jar
  endpoint: service:jmx:rmi:///jndi/rmi://127.0.0.1:2020/jmxrmi
  target_system: jvm,kafka,kafka-consumer,kafka-producer
  collection_interval: 10s
  log_level: info
  resource_attributes:
    broker.name: broker1
jmx/2:
  # configure the path where you installed opentelemetry-jmx-metrics receiver
  jar_path: ${PWD}/opentelemetry-jmx-metrics.jar
  endpoint: service:jmx:rmi:///jndi/rmi://127.0.0.1:2021/jmxrmi
  target_system: jvm,kafka,kafka-consumer,kafka-producer
  collection_interval: 10s
  log_level: info
  resource_attributes:
    broker.name: broker2
```

Kafka OTel Setup


```yaml
exporters:
  otlp:
    # configure the grpc endpoint for signoz otel collector
    endpoint: "127.0.0.1:65118"
    tls:
      insecure: true
  debug:
    verbosity: detailed

service:
  pipelines:
    traces:
      receivers: [otlp]
      exporters: [otlp]
    logs:
      receivers: [otlp]
      exporters: [otlp]
    metrics:
      receivers: [otlp, kafkametrics]
      exporters: [otlp, debug]
```

Distributed Tracing



Distributed Tracing


```
InstrumentationScope io.opentelemetry.kafka-clients-0.11 2.4.0-alpha
Span #0
    Trace ID       : d373467a29492793345307e726fb1404
    Parent ID      : ee72a4415b8d6574
    ID             : 9acfab51e1b25826
    Name           : topic1 process
    Kind           : Consumer
    Start time     : 2024-05-21 15:16:47.166954 +0000 UTC
    End time       : 2024-05-21 15:16:47.167983166 +0000 UTC
    Status code    : Unset
    Status message :
Attributes:
     -> messaging.kafka.message.offset: Int(4)
     -> messaging.kafka.consumer.group: Str(my-consumer-group)
     -> messaging.kafka.message.key: Str(MY_KEY)
     -> messaging.system: Str(kafka)
     -> messaging.destination.partition.id: Str(0)
     -> kafka.record.queue_time_ms: Int(30)
     -> messaging.destination.name: Str(topic1)
     -> messaging.operation: Str(process)
     -> thread.id: Int(39)
     -> messaging.client_id: Str(consumer-my-consumer-group-1)
     -> thread.name: Str(Thread-2)
     -> messaging.message.body.size: Int(10)
```

Kafka Client-Native Metrics

Name	Instrument Type	Unit (UCUM)	Description
`messaging.client.operation.duration`	Histogram	`s`	Duration of messaging operation initiated by a producer or consumer client.

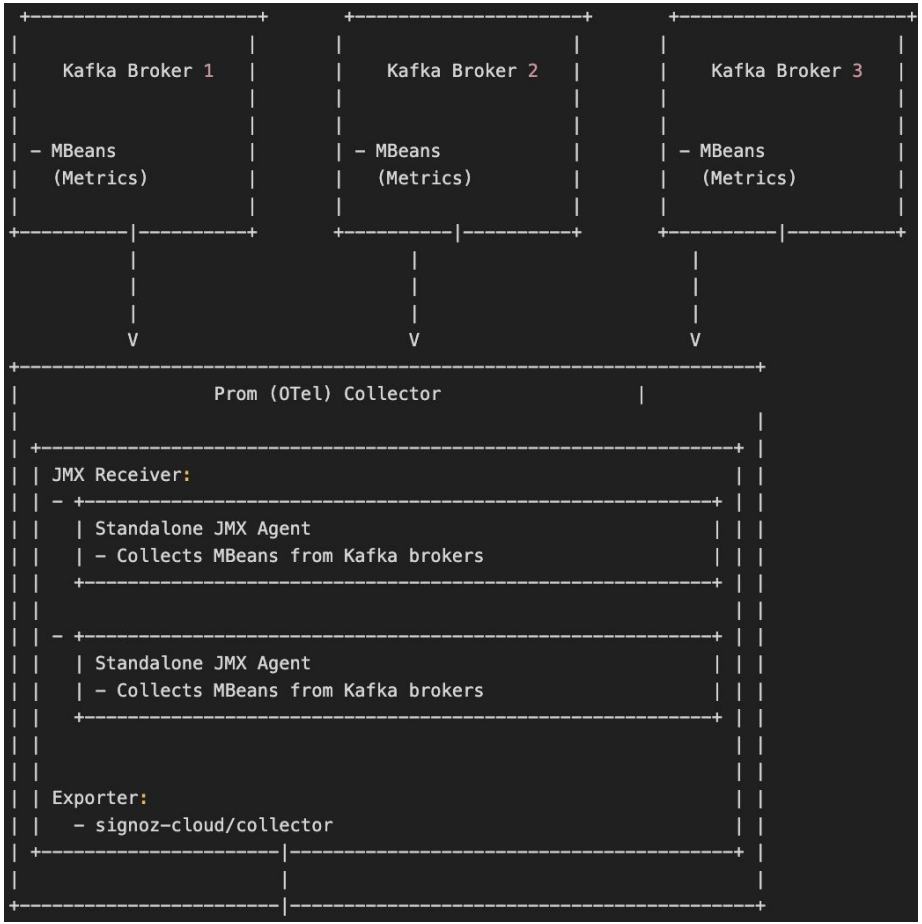
Attribute	Type	Description	Examples
`error.type`	string	Describes a class of error the operation ended with.	`amqp:decode-error`; `KAFKA_STORAGE_ERROR`; `channel-error`
`messaging.consumer.group.name`	string	The name of the consumer group with which a consumer is associated.	`my-group`; `indexer`
`messaging.destination.name`	string	The message destination name.	`MyQueue`; `MyTopic`
`messaging.destination.partition.id`	string	The identifier of the partition messages are sent to or received from, unique within the `messaging.destination.name`.	`1`
`messaging.operation.name`	string	The system-specific name of the messaging operation.	`send`; `receive`; `ack`
`messaging.operation.type`	string	A string identifying the type of the messaging operation.	`create`; `send`; `receive`
`messaging.system`	string	The messaging system as identified by the client instrumentation.	`activemq`; `aws_sqs`; `eventgrid`
`server.address`	string	Server domain name if available without reverse DNS lookup; otherwise, IP address or Unix domain socket name.	`example.com`; `10.1.2.80`; `/tmp/my.sock`
`server.port`	int	Server port number.	`80`; `8080`; `443`

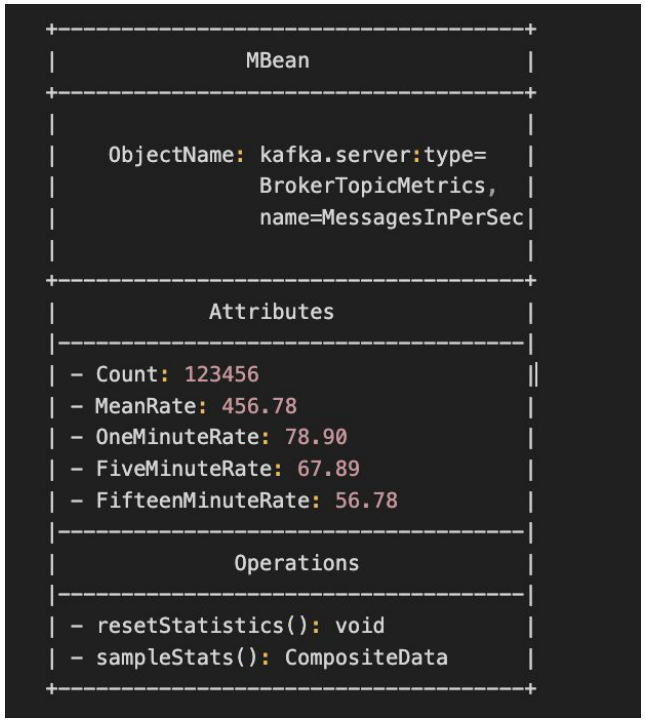
- messaging.operation.name
- messaging.system
- messaging.batch.message_count
- messaging.consumer.group.name
- messaging.destination.anonymous
- messaging.destination.name
- messaging.destination.subscription.name
- messaging.destination.template
- messaging.destination.temporary
- messaging.operation.type
- messaging.client.id
- messaging.destination.partition.id
- messaging.message.conversation_id
- messaging.message.id
- messaging.message.body.size
- messaging.message.envelope.size

Correlations from From Client spans

- Correlation with metrics based on Span Attributes
- See the time a message spent in kafka
- Can provide deeper visibility into your async systems

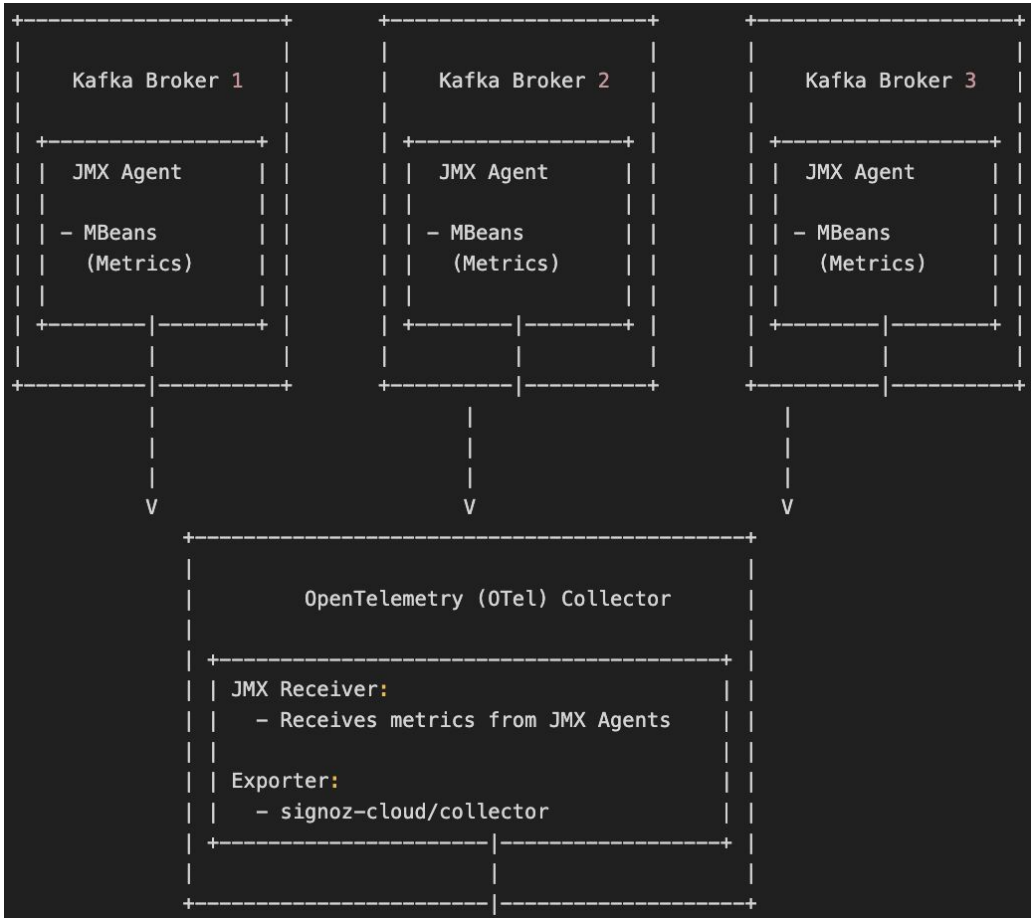
How to collect broker metrics collection?





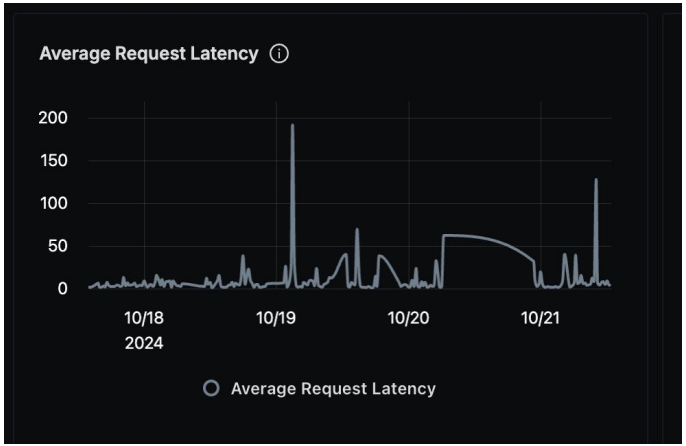
How to collect broker metrics collection?

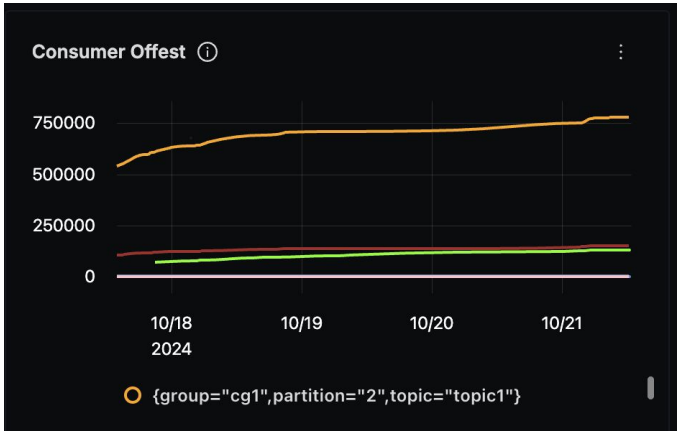
How to collect broker metrics collection?

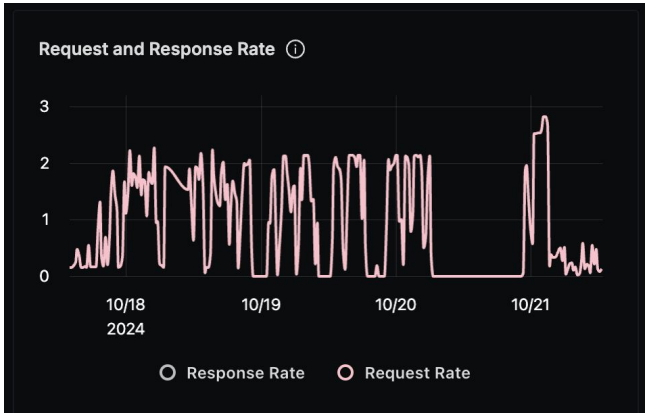


What's missing with the metrics?

Kafka Broker-Native Metrics







Demo

Try out the correlation!

https://signoz.io/docs/messaging-queues/kafka/

https://github.com/SigNoz/kafka-opentelemetry-instrumentation

Demo

- Participate in OTel Messaging Semantic Conventions
  - Define new metrics, add attributes or conventions for new systems
- Improve existing instrumentations
  - Available in Java, Python, .NET, node.js, and other languages
- Ask for native instrumentation in your favorite Kafka client library

KubeCon | CloudNativeCon

North America 2024

Thanks for joining!

Questions?

We appreciate your feedback!

SigNoz