

# Tecnologias Mensageria

Prof. Rodrigo Macedo

# Escopo do Curso

- RabbitMQ
- ActiveMQ
- Kafka
- Questão de Concursos



# ActiveMQ

- Apache ActiveMQ é um message broker de código-fonte aberto escrito em Java, juntamente com um cliente completo de Java Message Service (JMS).
- Ele fornece "recursos empresariais", ou seja, promove a comunicação de mais de um cliente ou servidor.
- A comunicação é gerida com recursos como clusters computacionais e a capacidade de utilizar qualquer banco de dados como um fornecedor de persistência JMS, além de memória virtual, cache, e persistência de logs.



# ActiveMQ

- O projeto ActiveMQ foi originalmente criado por seus fundadores da LogicBlaze em 2004, como um message broker de código aberto, hospedado pela CodeHaus.
- O código e a marca registrada ActiveMQ foram doados à Apache Software Foundation em 2007, sendo que os fundadores continuaram a desenvolver o código-base com a comunidade estendida Apache.
- ActiveMQ emprega vários modos para alta disponibilidade, incluindo ambos mecanismos de bloqueio de sistema de arquivos e de banco de dados a nível de linha, compartilhamento de armazenamento de persistência, através de um sistema de arquivos compartilhado, ou replicação real usando o Apache ZooKeeper.



# ActiveMQ

- No ramo empresarial, o ActiveMQ é celebrado por sua flexibilidade de configuração, e o seu suporte para um número relativamente grande de protocolos de transporte, incluindo OpenWire, Stomp, MQTT, AMQP, REST e WebSockets.
- ActiveMQ é usado em implementações de barramento de serviço corporativo, tais como o Apache ServiceMix e Mule. Outros projetos usando ActiveMQ incluem o Apache Camel e Apache CXF em projetos de infra-estrutura SOA.
- É escrito em Java e funciona de forma multiplataforma. Mantido pela Apache Software Foundation.
- Possui duas versões: Classic e Artemis. A versão Artemis, Arquitetura sem bloqueio de alto desempenho para a próxima geração de aplicativos de mensagens.



# ActiveMQ

- A integração do Apache ActiveMQ grava registros e coleta métricas de uso e mensagens do armazenamento. As métricas de armazenamento incluem uso de memória e disco.
- As métricas incluem o número de mensagens em espera, o tempo médio de espera e as mensagens expiradas.
- O ActiveMQ grava registros em syslog, que é capturado pelo agente de operações por padrão. Não é necessária outra configuração do usuário.
- Para ingerir métricas do ActiveMQ, é necessário criar receptores para as métricas que o ActiveMQ produz e, em seguida, criar um pipeline para os novos receptores.



# ActiveMQ

- Acessamos o site: <http://activemq.apache.org/>
- Na sessão Download, podemos baixar a versão mais atual ou em Getting past release podemos baixar uma versão mais antiga para ambiente windows ou linux.
- Iremos baixar no nosso exemplo a versão 5.12.0 para windows, e extrair o arquivo para a pasta C:\jms\apache-activemq-5.12.0 você pode nomear a sua pasta como quiser, contanto que não tenham espaços no caminho do seu diretório para evitar problemas futuros.
- Pelo prompt de comando do windows ou pelo terminal do seu linux navegamos no diretório até a pasta bin e executamos o comando para startar o servidor.

Windows: `activemq.bat start`


Linux: `sh activemq start`

- Acessar em: <http://localhost:8161/> e <http://localhost:8161/admin>





# ActiveMQ - Configuração

- Para configurar um receptor para suas métricas do activemq, especifique os campos a seguir:

Campo	Padrão	Descrição
collection_interval	60s	Um valor de <a href="#">time.Duration</a>  , como 30s ou 5m.
endpoint	http://localhost:1099	O URL do nó a ser monitorado.
password		Senha configurada se o JMX estiver configurado para exigir autenticação.
type		Este valor precisa ser <code>activemq</code> .
username		O nome de usuário configurado se o JMX estiver configurado para exigir autenticação.



# ActiveMQ



Home | Queues | Topics | Subscribers | Connections | Network | Scheduled | Send

Support

Topic Name

## Topics

Name ↑	Number Of Consumers	Messages Enqueued	Messages Dequeued	Operations
#Amigos do Trabalho	0	0	0	<a href="#">Send To Active Subscribers</a> <a href="#">Active Producers</a> <a href="#">Delete</a>
#folha	0	0	0	<a href="#">Send To Active Subscribers</a> <a href="#">Active Producers</a> <a href="#">Delete</a>
#hcm	0	0	0	<a href="#">Send To Active Subscribers</a> <a href="#">Active Producers</a> <a href="#">Delete</a>
@alicia	0	0	0	<a href="#">Send To Active Subscribers</a> <a href="#">Active Producers</a> <a href="#">Delete</a>
@bianca	0	0	0	<a href="#">Send To Active Subscribers</a> <a href="#">Active Producers</a> <a href="#">Delete</a>
@bruna	0	0	0	<a href="#">Send To Active Subscribers</a> <a href="#">Active Producers</a> <a href="#">Delete</a>
@gabi	0	0	0	<a href="#">Send To Active Subscribers</a> <a href="#">Active Producers</a> <a href="#">Delete</a>
				<a href="#">Send To Active Subscribers</a>

[Queue Views](#)

- [Graph](#)
- [XML](#)

[Topic Views](#)

- [XML](#)

[Subscribers Views](#)

- [XML](#)

[Useful Links](#)

- [Documentation](#)
- [FAQ](#)
- [Downloads](#)
- [Forums](#)

# RabbitMQ

- O RabbitMQ é um software de mensagens com código aberto, que implementou o protocolo "Advanced Message Queuing Protocol" (AMQP), que foi estendido com uma arquitetura de plug-in para suportar o protocolo "Streaming Text Oriented Messaging Protocol" (STOMP), o MQTT entre outros protocolos.
- É escrito em Erlang (linguagem de programação) e seu código fonte é publicado sob a licença pública Mozilla. Foi lançado inicialmente pela Rabbit Technologies em 2007.
- O RabbitMQ é leve e fácil de implantar local ou em Cloud, além de atender requisitos de alta disponibilidade e alta escala.



# RabbitMQ

- Chamamos o RabbitMQ de um Message Broker(Intermediador de mensagens).
- Quando trabalhamos com o RabbitMQ a nossa aplicação envia as mensagens para uma Exchange e ele se encarrega de organizar as mensagens nas filas.
- **Exchange:** Uma Exchange é quem vai definir para qual fila as mensagens recebidas vão. Fazendo uma analogia podemos dizer que uma Exchange é como um roteador de internet. No RabbitMQ a sua aplicação somente envia mensagens para uma Exchange, nunca para uma fila diretamente.
- O RabbitMQ possui quatro tipos de Exchange, são elas **direct**, **fanout**, **headers** e **topic** e uma Exchange coringa chamada AMQP default.

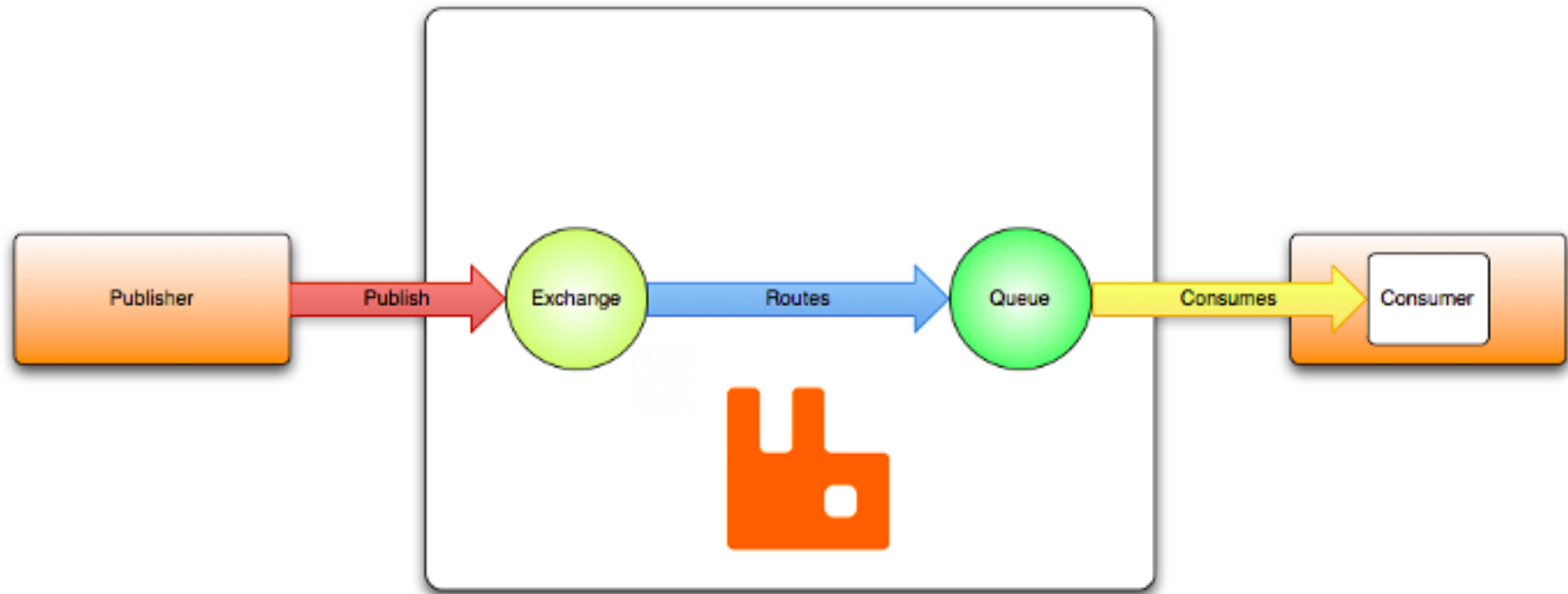
# RabbitMQ - Exchanges

1. **Direct:** O Direct Exchange encaminha as mensagens procurando por um binding key igual ao routing key fornecido. Vale salientar que se o exchange não consegue encaminhar a mensagem para uma fila, ela será descartada.
2. **Topic:** Topic Exchange funciona de uma forma parecida ao Direct, porém o binding key é um tipo de “expressão regular” aplicada sobre o routing key. Para ser mais fácil de entender, poderíamos fazer uma analogia com as tags do post.
3. **Fanout:** O Fanout Exchange encaminha para todas as filas com binding nele, desconsiderando routing key.
4. **Headers:** O Headers Exchange usa os valores do header da mensagem para fazer o encaminhamento, ignorando o routing key. É parecido ao Topic mas permite um controle mais fino.

# RabbitMQ

- No exemplo a seguir, é possível observar um exemplo básico de publicação de mensagem no exchanges, roteamento para uma queue e recepção da mensagem por um consumidor.

"Hello, world" example routing



# RabbitMQ - Instalação

- Uma das formas de instalar o RabbitMQ é por meio de Docker. Você pode executar o comando do Docker abaixo para subir um container do RabbitMQ.
- `docker run -d -it --rm --name rabbitmq -p 5672:5672 -p 15672:15672 rabbitmq:3-management`
- Para acessar o admin, digite no navegador: `http://localhost:15672`
- O usuário e senha padrão para você acessar é guest.




Username:

Password:

Login

# RabbitMQ - Instalação

 RabbitMQ™  
RabbitMQ 3.8.4 Erlang 23.0.2

Refreshed 2020-05-29 11:25:11 Refresh every 5 seconds ▾

Virtual host All ▾

Cluster rabbit@ce21300cfa8

User guest Log out

Overview

Connections Channels Exchanges Queues Admin

## Overview

▼ Totals

Queued messages last minute ?

Currently idle

Message rates last minute ?

Currently idle

Global counts ?

Connections: 0 Channels: 0 Exchanges: 7 Queues: 0 Consumers: 0

▼ Nodes

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Info	Reset stats	+/-
rabbit@ce21300cfa8	33 1048576 available	0 943629 available	441 1048576 available	102MiB 6.1GiB high watermark	175GiB 48MiB low watermark	8m 0s	basic disc 1 rss	This node All nodes	

▼ Churn statistics

Connection operations last minute ?

# RabbitMQ - Exemplo

- Para realizar os testes vamos usar Pika uma biblioteca cliente do RabbitMQ e será disponibilizado os códigos em python como exemplos de produtor e consumidor.

```
import pika
import time

connection = pika.BlockingConnection(
    pika.ConnectionParameters('localhost')
)

channel = connection.channel()
channel.queue_declare(queue='hello')

messages = [
    "Primeira mensagem",
    "Segunda mensagem",
    "Terceira mensagem"
]

for message in messages:
    channel.basic_publish(
        exchange='',
        routing_key='hello',
        body=message
    )

    print(" [x] Enviada '" + message + "'")
    time.sleep(1)

connection.close()
```



# RabbitMQ - Exemplo

- Para realizar os testes vamos usar Pika uma biblioteca cliente do RabbitMQ e será disponibilizado os códigos em python como exemplos de produtor e consumidor.

```
import pika

def callback(ch, method, properties, body):
    print(" [x] Received %r" % body)

connection = pika.BlockingConnection(
    pika.ConnectionParameters('localhost')
)

channel = connection.channel()
channel.queue_declare(queue='hello')
channel.basic_consume(
    queue='hello',
    auto_ack=True,
    on_message_callback=callback
)

print(' [*] Waiting for messages. To exit press CTRL+C')

channel.start_consuming()
```

# RabbitMQ - Quando usar

- Você pode usar o RabbitMQ sempre que você precisar fazer a comunicação entre aplicações.
- Quando não precisa de uma resposta imediata. No caso **requisições assíncronas**, Operações como enviar um e-mail, SMS são ótimos exemplos de requisições a onde o usuário só precisa saber que o email ou o SMS vai ser enviado, não precisa saber exatamente quando isso aconteceu.
- Outro caso é quando você sabe que para processar a requisição que o usuário solicitou vai demorar. Aí você não quer deixar ele esperando, você pode usar o RabbitMQ pegar a requisição enviar para uma fila para ser processada e para o usuários que assim que for processado ele vai receber uma notificação do sistema.

# RabbitMQ - Quando não usar

- Nos casos em que você precisa fazer uma comunicação síncrona a onde o seu usuário precisa saber na hora se sua solicitação foi resolvido ou não.
- Até existe um Pattern(padrão) chamado RPC(Remote Procedure Call) para fazer comunicações síncronos com AMQP consequentemente com o RabbitMQ mais cabe a você fazer a implementação desse pattern e fazer isso pode acabar sendo mais custoso do simplesmente usar o HTTP.
- Em geral, nesses casos, é mais interessante utilizar uma outra estratégia, do que utilizar um broker de mensageria, como o RabbitMQ.

# Apache Kafka

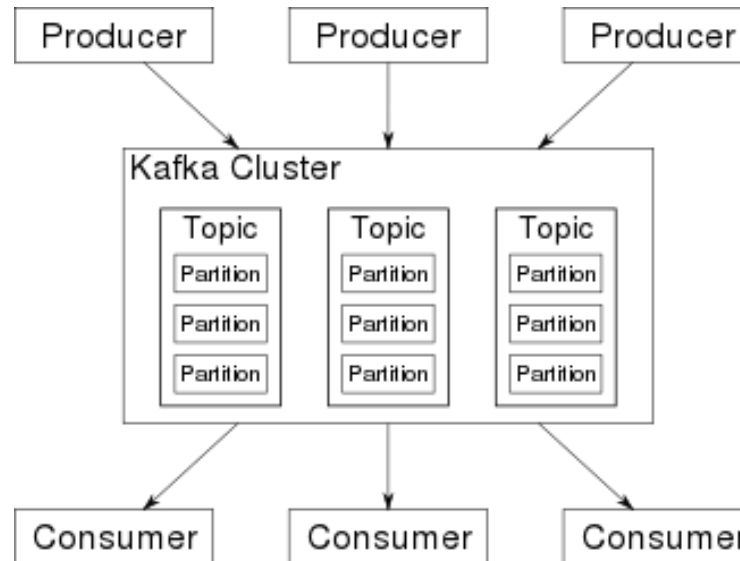
- Apache Kafka é uma plataforma open-source de processamento de streams desenvolvida pela Apache Software Foundation, escrita em Scala e Java.
- O projeto tem como objetivo fornecer uma plataforma unificada, de alta capacidade e baixa latência para tratamento de dados em tempo real.
- Sua camada de armazenamento é, essencialmente, uma "fila de mensagens de publishers/subscribers que trabalha de forma escalável projetada como um log de transações distribuído", tornando-o altamente valioso para infra-estruturas corporativas que processam transmissão de dados.
- Além disso, Kafka se conecta a sistemas externos (para importação/exportação de dados) através do Kafka Connect e ainda fornece o Kafka Streams, uma biblioteca Java de processamento de fluxos.

# Apache Kafka

- O Apache Kafka é baseado no commit log, ele permite que os usuários inscrevam-se e publiquem dados para qualquer número de sistemas ou aplicações em tempo real.
- Exemplos de aplicações incluem o gerenciamento de correspondência entre passageiros e um condutor no Uber, fornecer análises e métricas em tempo real, além de realizar inúmeros serviços em tempo real para todo o LinkedIn.
- O Kafka armazena mensagens por chave-valor que provêm de inúmeros processos arbitrários chamados de "produtores". Os dados podem, assim, ser particionados em diferentes "partições" e diferentes "tópicos".
- Dentro de uma partição, as mensagens são estritamente ordenadas pela sua posição e indexados e armazenados juntamente com suas data e hora.
- Outros processos chamados de "consumidores", podem ler mensagens de partições.

# Apache Kafka

- São quatro as principais APIs no Kafka:
  1. **Producer API**– Permite que as aplicações publiquem fluxos de dados.
  2. **Consumer API**– Permite que a aplicação assine tópicos e processe o fluxo de dados.
  3. **Connector API**– Executa as API's de produtor e consumidor que podem vincular os tópicos para os aplicativos existentes.
  4. **Streams API**– Esta API converte os fluxos de entrada para um fluxo de saída e produz o resultado.



# Apache Kafka

- Kafka é diferente dos sistemas tradicionais de mensageria. Conceitualmente, ele é um sistema de log distribuído.
- Isso significa que as mensagens enviadas ao Kafka são replicadas entre os nodos do Cluster e salvas de modo sequencial, o que garante que ao lermos os registros eles serão entregues na mesma ordem na qual foram enviados.
- Um dos principais fatores que distingue o Kafka de outros serviços de mensageria é a tolerância a falhas e alta disponibilidade de dados que ele fornece. Isso se dá por meio da replicação das informações.
- É possível configurar o Kafka para copiar esses dados para outros brokers, mantendo-os em outros locais.
- Uma das configurações comuns de produção para atingir esse objetivo é o fator de replicação de 3. Dessa forma, haverá três cópias desses materiais em outras partições.

# Apache Kafka - Componentes

-> **Log:** Um log pode ser descrito como uma sequência temporal de mensagens, onde as novas mensagens sempre são adicionadas no final do log. Desta forma, uma mensagem enviada em  $t_0$  sempre estará posicionada antes de uma mensagem enviada em  $t_1$ .

- Cada mensagem dentro do log possui algumas informações:

1. **Timestamp:** data-hora da inserção

2. **Offset:** índice da mensagem na partição

3. **Key:** chave da mensagem

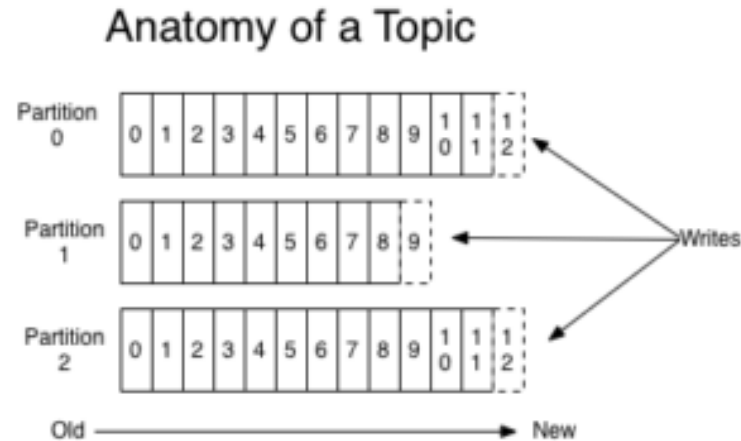
4. **Value:** a mensagem propriamente dita chamado de payload

- Todas as mensagens dentro de uma partição serão um conjunto chave / valor.



# Apache Kafka - Componentes

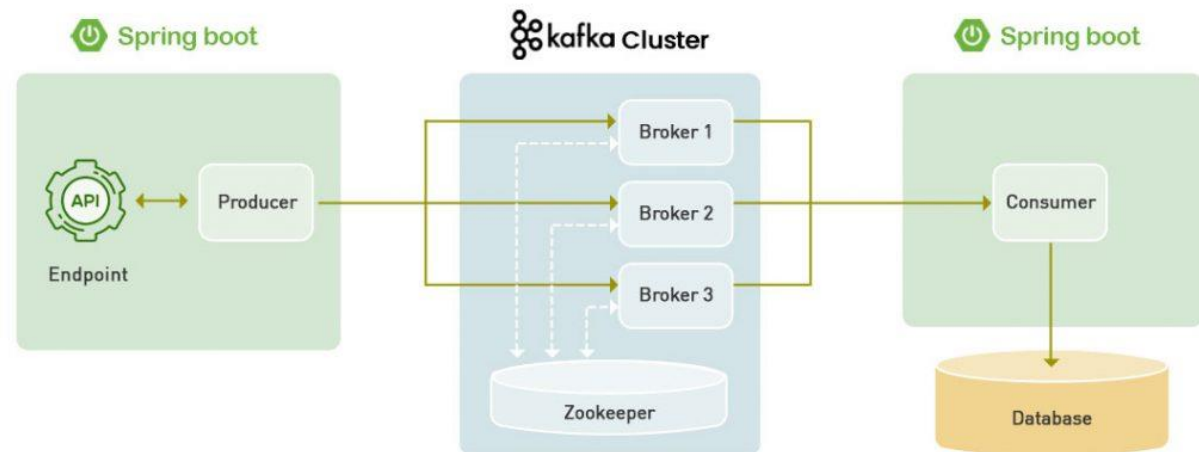
-> **Partição:** Kafka utiliza o conceito de tópico onde as mensagens são agrupadas por uma classificação lógica. Por exemplo, se um tópico chamado **salarios\_funcionarios** existir, esperamos que dentro dele estejam armazenadas mensagens desta categoria. Internamente, um tópico é dividido em partições.



- Este tópico possui 3 partições, cada uma com um número específico de mensagens e totalmente independentes.
- Cada partição é um log isolado. Com isso podemos concluir que a ordem das mensagens só pode ser garantida dentro de uma partição.

# Apache Kafka - Componentes

- > **Broker:** Um broker é o componente responsável por receber as requisições de producers e consumers, armazenar as mensagens e executar a replicação das mesmas.
- Os brokers são gerenciados por outro componente o **zookeeper**. Este componente é bastante utilizado para controlar os diferentes integrantes de um cluster.
  - Além das tarefas descritas acima, os brokers também realizam outras tarefas, como gerenciar os líderes de cada partição, realizar a limpeza de dados ou a compactação das mensagens.



## Questão Concursos

**Q1) [CEBRASPE SERPRO 2021]** A respeito do software Kafka, julgue o item a seguir.

É possível fazer uso da API consumer para publicar eventos em tópicos Kafka.

**Q2) [CEBRASPE SEFAZ-CE 2021]** Julgue o próximo item, relativos ao Apache Kafka e ao Kubernetes.

O Apache Kafka provê serviço de mensageria e integração de dados, de forma assíncrona, em que produtores e consumidores ficam desacoplados e agnósticos entre si.

## Questão Concursos

**Q1) [CEBRASPE SERPRO 2021]** A respeito do software Kafka, julgue o item a seguir.

É possível fazer uso da API consumer para publicar eventos em tópicos Kafka. **ERRADO.**

**Q2) [CEBRASPE SEFAZ-CE 2021]** Julgue o próximo item, relativos ao Apache Kafka e ao Kubernetes.

O Apache Kafka provê serviço de mensageria e integração de dados, de forma assíncrona, em que produtores e consumidores ficam desacoplados e agnósticos entre si. **CERTO.**