RabbitMQ

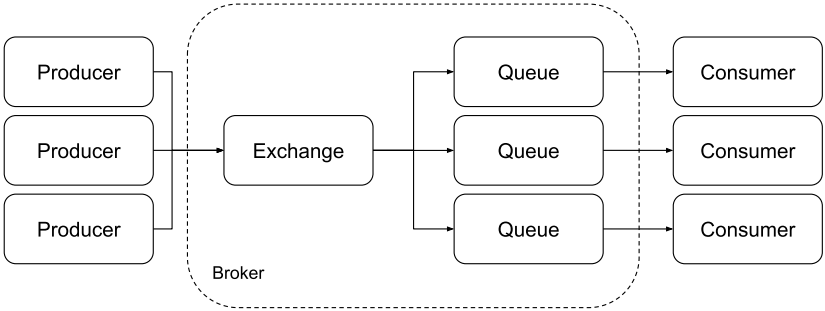
note by umluizlima

> RabbitMQ is a message broker that's useful for asynchronous communication between applications.

Content

RabbitMQ is a message broker that implements the **Advanced Message Queueing Protocol** (AMQP), which standardizes messaging using producers, a broker, and consumers.

Message flow



1. A **Producer** publishes a message containing a **routing key** to an Exchange;
2. An **Exchange** has bindings with binding keys to one or several queues and distributes messages based on its type;
3. A **Consumer** polls its queue to retrieve and process messages;

Message consumption can be manually acknowledged by its consumer to prevent it from being lost on application failure, and messages that cannot be routed may be returned to its publishers, dropped, or placed in a dead letter queue.

Exchanges, Queues, and bindings can be referred to as **AMQP entities**.

AMQP is a **programmable protocol**, so its entities can be declared by the applications that connect to the Broker. This means that it is possible to **declare all entities and their settings programmatically**.

Queues and Exchanges have some properties in common, such as:

- Name;
- Durable, tell if the entity will survive a broker restart;
- Auto-delete, the entity is deleted when no other entities are connected or bound to it;

Bindings are rules used by exchanges to route messages. The **routing key** attribute is optional to bindings and is only used on specific exchange types. Its purpose is to allow for certain messages published to an exchange to be routed, like a **filter**.

Exchange types

- **Fanout** ignores the message's routing key and sends it to all queues;
- **Direct** sends the message to the queue where `routing key == binding key`;
- **Topic** sends the message on a partial match of keys;
- **Header** uses the message header instead of its routing key;
- **Default** (nameless) sends message to queue where `routing key == queue name`;

Example

```
$ pip install pika
$ python
>>> import pika
>>> connection =
pika.BlockingConnection(pika.ConnectionParamete
rs('localhost'))
>>> channel = connection.channel()
```

The exchange, queue and binding between them can be created programmatically:

```
>>>
channel.exchange_declare(exchange='my_exchange'
, exchange_type='fanout')
>>> result =
channel.queue_declare(queue='my_queue',
durable=True)
>>> queue_name = result.method.queue
>>> channel.queue_bind(exchange='my_exchange',
queue=queue_name)
```

The exchange must be passed to publish a message. So does the routing key, even if not used:

```
>>> message = "There's a snake in my boot!"
>>>
channel.basic_publish(exchange='my_exchange',
routing_key='', body=message)
```

Consumption can be triggered by specifying the queue and passing a callback function:

```
>>> def callback(ch, method, properties, body):
>>>     ...
>>> channel.basic_consume(queue=queue_name,
on_message_callback=callback, auto_ack=True)
```

References

- RabbitMQ in 5 Minutes
- AMQP Concepts
- RabbitMQ Tutorials