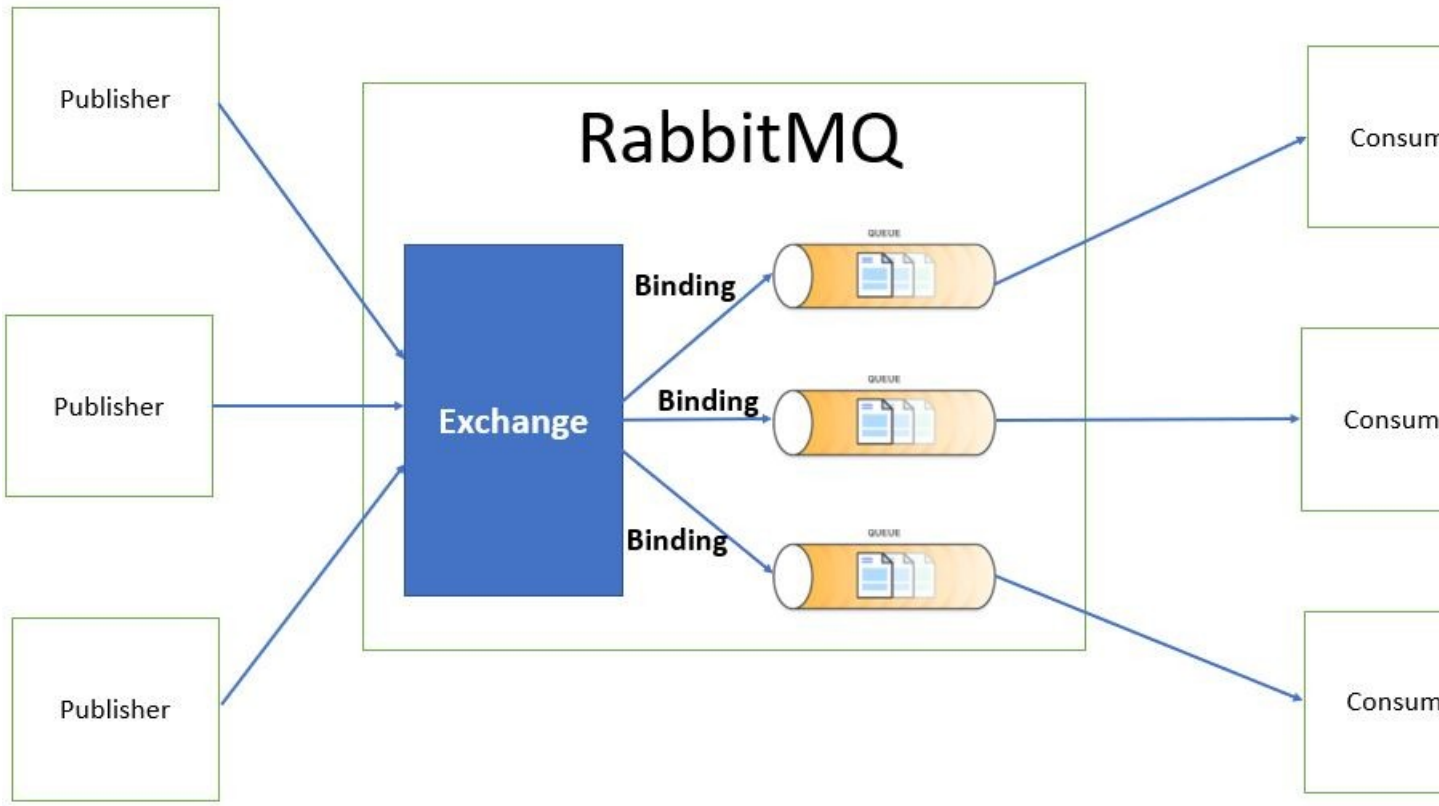
RabbitMQ Messaging Flow -

When using RabbitMQ the publisher never directly sends a message to a queue. Instead, the publisher sends messages to an exchange. Exchange is responsible for sending the message to an appropriate queue based on routing keys, bindings and header attributes. Exchanges are message routing agents which we can define and bindings are what connects the exchanges to queues. So in all our examples we will be creating first a Queue and Exchange, then bind them together.

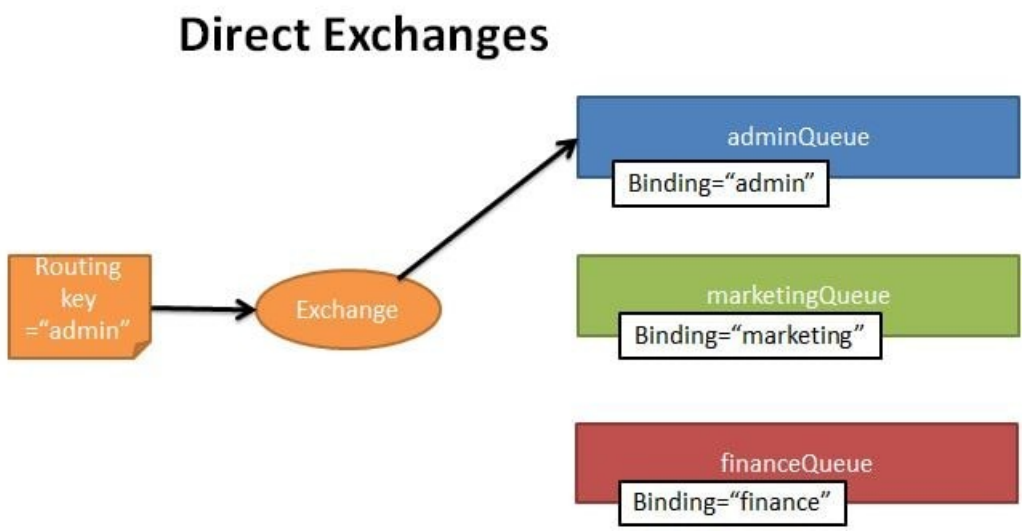


With RabbitMQ we have the following types of Exchanges-

- Direct Exchange
- Fanout Exchange
- Topic Exchange
- Header Exchange

Direct Exchange

Based on the routing key a message is sent to the queue having the same routing key specified in the binding rule. The routing key of exchange and the binding queue have to be an exact match. A message is sent to exactly one queue.

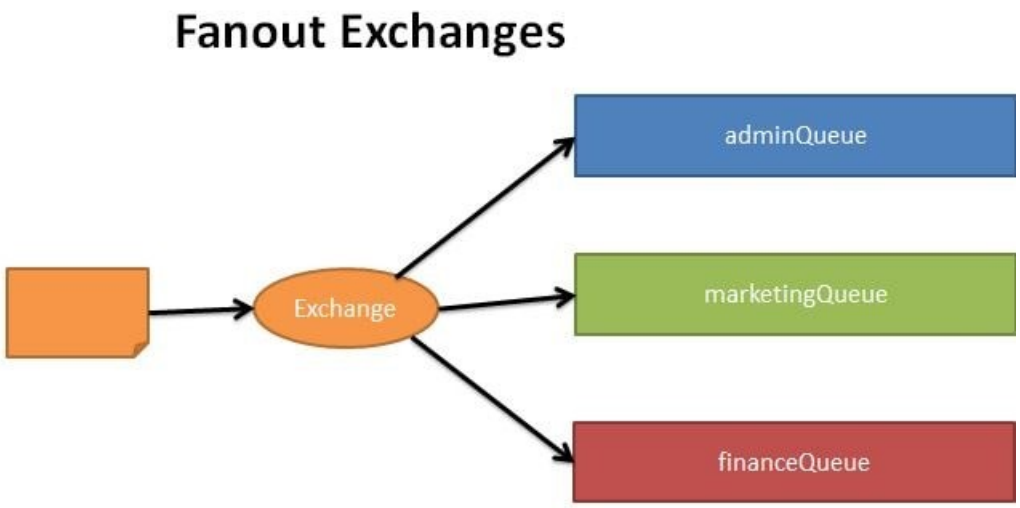


Run the Spring Boot Application

- We send the message using the url
  - **http://localhost:8080/javainuse-rabbitmq/direct/producer?exchangeName=direct-exchange&routingKey=admin&messageData=HelloWorldJavaInUse**
  we will be specifying the following
    - exchange name= "direct-exchange"
    - routing key ="admin"
    - message to sent to queue = "HelloWorldJavaInUse"

Fanout Exchange

The message is routed to all the available bounded queues. The routing key if provided is completely ignored. So this is a kind of publish-subscribe design pattern.



Modify the RabbitMQFanoutConfig as follows-

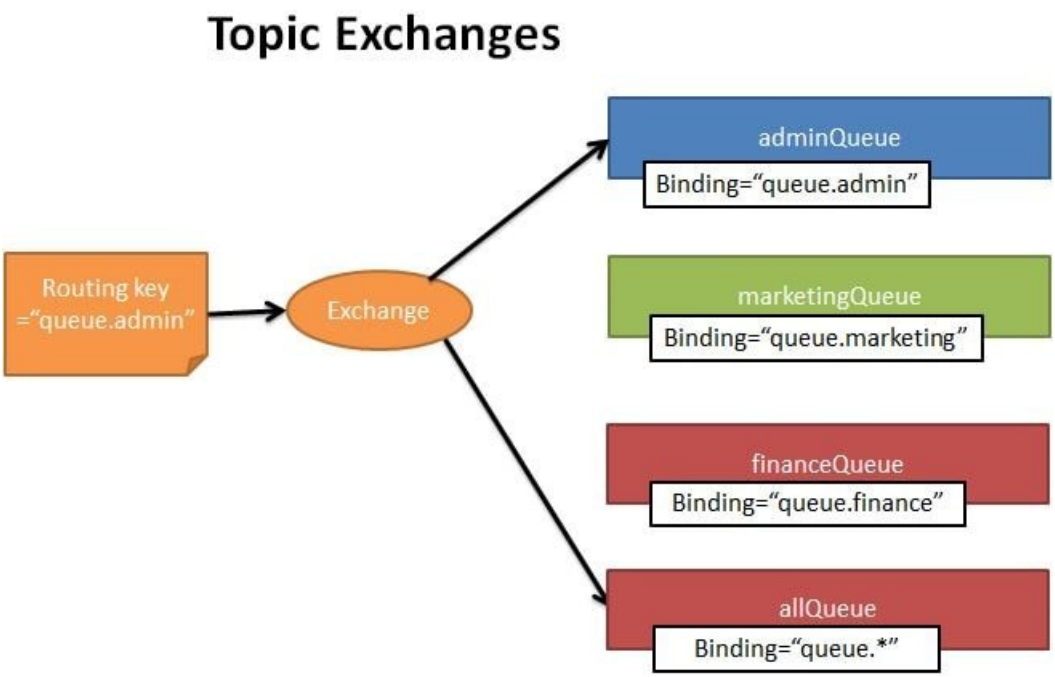
- Create Queues named - marketingQueue, adminQueue, financeQueue
- Create a FanoutExchange named - fanout-exchange
- Create Bindings for each of the queue with the FanoutExchange. Also as this is a fanout exchange we do not need to specify a binding key.

Run the Spring Boot Application

- We send the message using the url
  - **http://localhost:8080/javainuse-rabbitmq/fanout/producer?exchangeName=fanout-exchange&messageData=HelloWorldJavaInUse**
    - exchange name= "fanout-exchange"
    - message to sent to queue = "HelloWorldJavaInUse"
- We do not need to specify the routing key here as message is published to all the queues. The message is sent to the admin queue.We get the web

Topic Exchange

Here again the routing key is made use of. But unlike in direct exchange type, here the routing key of the exchange and the bound queues should not necessarily be an exact match. Using regular expressions like wildcard we can send the exchange to multiple bound queues.



Modify the RabbitMQTopicConfig as follows-

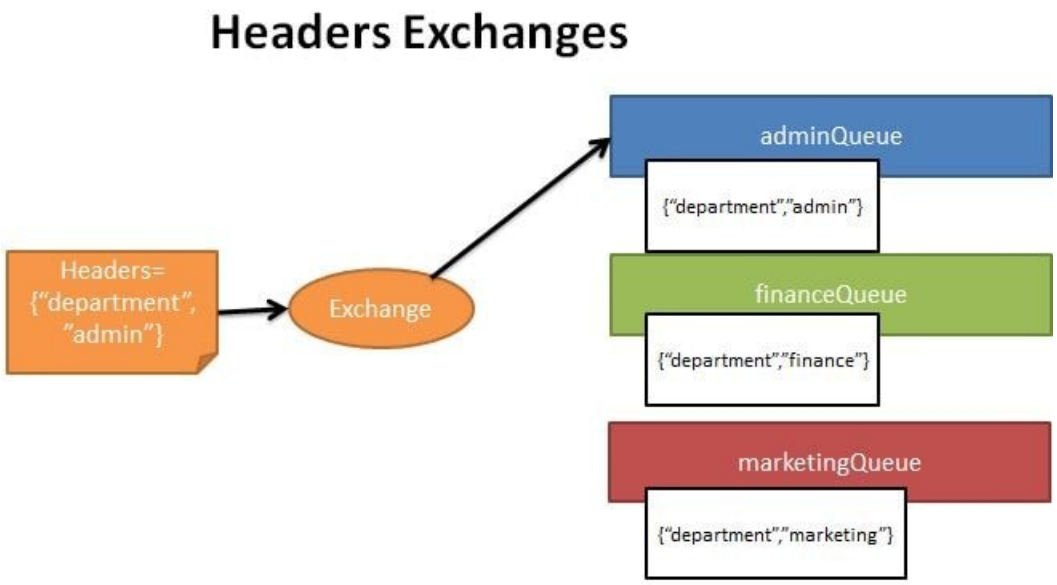
- Create Queues named - marketingQueue, adminQueue, financeQueue and allQueue
- Create a TopicExchange named - topic-exchange
- Create Bindings for each of the queue with the TopicExchange. We specify routing key for each binding. Also for allQueue binding we specify the binding key with wildcard.

Run the Spring Boot Application

- We send the message using the url
  - **http://localhost:8080//javainuse-rabbitmq/topic/producer?exchangeName=topic-exchange&routingKey=queue.admin&messageData=HelloWorldJavaInUse**
    - exchange name= "topic-exchange"
    - routing key ="queue.admin"
    - message to be sent to queue = "HelloWorldJavaInUse"

Header Exchange

In this type of exchange the routing queue is selected based on the criteria specified in the headers instead of the routing key. This is similar to topic exchange type, but here we can specify complex criteria for selecting routing queues.



Modify the RabbitMQHeaderConfig as follows-

- Create Queues named - marketingQueue, adminQueue, financeQueue
- Create a HeaderExchange named - header-exchange
- Create Bindings for each of the queue with the HeaderExchange. As this is a Header Exchange, instead of binding key we specify the criteria rules which should be present in the message header.

Run the Spring Boot Application

- We send the message using the url
  - **http://localhost:8080/javainuse-rabbitmq/header/producer?exchangeName=header-exchange&department=admin&messageData=HelloWorldJavaInUse**
    - exchange name= "header-exchange"
    - header key ="admin"
    - message to be sent to queue = "HelloWorldJavaInUse"