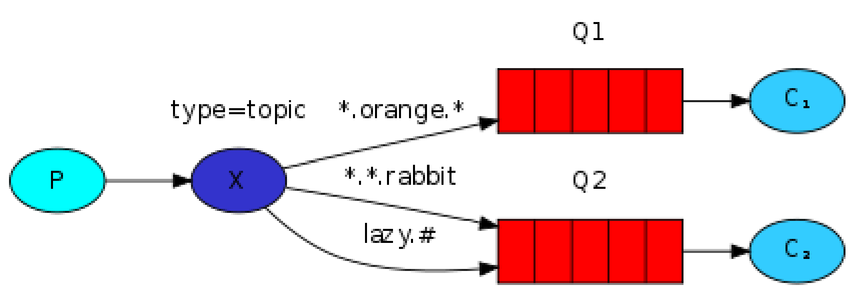
Python RabbitMQ主题交换机

在前面的代码中我们使用了直连交换机替代了扇型交换机，将只能盲目的广播消息改进为选择性的接收日志，尽管直连交换机能够改善我们的系统，但是它自身也有局限性，那就是没办法基于多个标准执行路由操作.

主题交换机:
发送到主题交换机(topic exchange)的消息携带的一个路由键必须是一个由.分隔开的词语列表，这些单词随便是什么都可以，但是最好是跟携带它们的消息有关系的词汇. 以下是几个推荐的例子: "stock.usd.nyse", "nyse.vmw", "quick.orange.rabbit". 词语的个数可以随意，但是不要超过255字节. 一个携带着特定路由键的消息会被主题交换机投递给与之相匹配的队列.
*(星号)用来表示一个单词.
#(井号)用来表示任意数量(零个或多个)单词.

如图所示:



在上图例子中，我们发送的所有消息都是用来描述小动物的. 发送的消息所携带的路由键是由三个单词所组成的，这三个单词被两个.分割开. 路由键里的第一个单词描述的是动物的手脚的利索程度，第二个单词是动物的颜色，第三个是动物的种类. 所以它看起来是这样的: <celerity>.<colour>.<species>

我们创建了三个绑定:
Q1的绑定键为 *.orange.*
Q2的绑定键为 *.*.rabbit 和 lazy.#

这三个绑定键被可以总结为:
Q1 对所有的桔黄色动物都感兴趣
Q2 则是对所有的兔子和所有懒惰的动物感兴趣

此时
quick.orange.rabbit的消息将会被分别投递给这两个队列
lazy.orange.elephant的消息也会被投递给这两个队列.
quick.orange.fox的消息会投递给第一个队列
lazy.brown.fox的消息会投递给第二个队列
lazy.pink.rabbit的消息只会被投递给第二个队列一次，即使它同时匹配第二个队列的两个绑定
quick.brown.fox的消息不会投递给任何一个队列

如果我们违反约定，发送了一个携带有一个单词或者四个单词("orange" or "quick.orange.male.rabbit")的消息时，发送的消息不会投递给任何一个队列而是会丢失掉.

示例代码(发送者):

```
In [ ]:
#coding:utf8
import pika, sys
cert = pika.PlainCredentials('rabbit', 'rabbit')
para = pika.ConnectionParameters('172.16.70.251', '5672', '/', cert)
connect = pika.BlockingConnection(para)
channel = connect.channel()
channel.exchange_declare(exchange="topic_logs",
                         exchange_type="topic")
topic = sys.argv[1]
message = sys.argv[2]
channel.basic_publish(exchange='topic_logs',
                      routing_key=topic,
                      body=message
                      )
print('message topic %s body %s send done!!' %(topic, message))
connect.close()
```

示例代码(接收者1):

```
In [ ]:
import pika, sys
import time
cert = pika.PlainCredentials('rabbit', 'rabbit')
para = pika.ConnectionParameters('172.16.70.251', '5672', '/', cert)
connect = pika.BlockingConnection(para)
channel = connect.channel()
channel.exchange_declare(exchange="topic_logs", exchange_type="topic")
result = channel.queue_declare(exclusive=True, queue='')
queue_name = result.method.queue
print(queue_name)
levels = ["*.*.rabbit", "lazy.#"]
for level in levels:
   print(level)
   channel.queue_bind(exchange='topic_logs',
                      queue=queue_name,
                      routing_key=level)

def callback(ch, method, prop, body):
   print('this is callback level is %s args is %s'%(method.routing_key, body))
channel.basic_consume(queue_name, callback, auto_ack=True)
channel.start_consuming()
```

示例代码(接受者2):

```
In [ ]:
import pika, sys
import time
cert = pika.PlainCredentials('rabbit', 'rabbit')
para = pika.ConnectionParameters('172.16.70.251', '5672', '/', cert)
connect = pika.BlockingConnection(para)
channel = connect.channel()
channel.exchange_declare(exchange="topic_logs", exchange_type="topic")
result = channel.queue_declare(exclusive=True, queue='')
queue_name = result.method.queue
levels = ["*.orange.*"]
for level in levels:
    print(level)
    channel.queue_bind(exchange='topic_logs',
                       queue=queue_name,
                       routing_key=level)

def callback(ch, method, prop, body):
   print('this is callback level %s args is %s'%(method.routing_key, body))
channel.basic_consume(queue_name, callback, auto_ack=True)
channel.start_consuming()
```