RabbitMQ - Sprawozdanie

Dawid Dębowski, 305290

Zadanie wykonałem samodzielnie

Zadanie 1a

Większą niezawodność zapewnia potwierdzenie po zakończeniu wiadomości. Dzięki temu w przypadku jakichś nieprzewidzianych okoliczności (jak wyłączenie / zresetowanie konsumenta), kolejka dalej pchnie wartość, która została wysłana.

Jeśli nie będziemy potwierdzać wiadomości ani po otrzymaniu, ani po przetworzeniu, wiadomość będzie wysłana jeszcze raz, nieważne, czy już przetworzyliśmy wiadomość, czy nie. Z przeprowadzonego przeze mnie doświadczenia wynika także, że jeśli zrestartujemy klienta po otrzymaniu wiadomości, wiadomość z kolejki także zostanie wysłana 2 razy.

Zadanie 1b

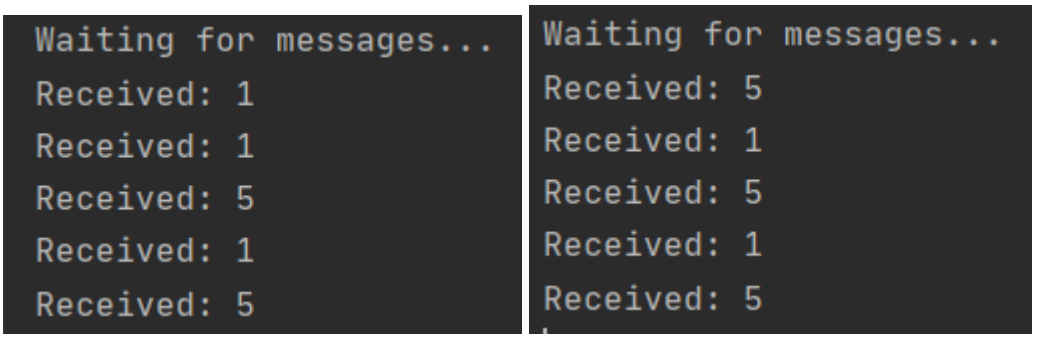
Po zmianie jedynie kodu Producenta, żeby przyjmował wiadomości do wysłania w nieskończoność oraz wpisaniu na zmianę ciągu 1,5, zaobserwowałem, że wszystkie wiadomości 1 zostały wypisane przez pierwszego Konsumenta, natomiast wszystkie wiadomości 5 - przez drugiego.

Output Konsumentów: (przed włączeniem obsługi QoS)

```
Z1 CONSUMER
Waiting for messages...
Received: 1
Received: 1
Received: 1
Received: 1
Received: 1
```

```
Z1 CONSUMER
Waiting for messages...
Received: 5
Received: 5
Received: 5
Received: 5
Received: 5
```

Output Konsumentów (po włączeniu QoS):



Jak widać, teraz Konsumenci "brali" wiadomości wtedy, kiedy byli wolni, nie na przemian tak, jak było to wcześniej.

Zadanie 2

W obydwóch doświadczeniach używać będę trzech konsumentów oraz jednego producenta. Producent otrzymuje za każdym razem klucz i wiadomość, następnie wysyła wiadomość zgodnie z zadanym kluczem. Klucze konsumentów są następujące: black.cat, blue.cat oraz black.* (klucz black.* pokaże różnicę między wysyłaniem Direct a wysyłaniem Topic).

Po wykonaniu eksperymentu na typie DIRECT okazało się (zgodnie z przewidywaniami), że żadna wiadomość nie dotarła do więcej niż jednego konsumenta. Aby sprawdzić poprawność rozwiązania, stworzyłem jeszcze jednego klienta z kluczem black.cat. Gdy ponownie wysłałem wiadomość z tym kluczem, doszła ona do konsumenta pierwszego oraz czwartego (tych, którzy mają klucze black). Jak widać, tryb DIRECT nie obsługuje topiców. Po wysłaniu wiadomości z kluczem black.cat, nie doszła ona do konsumenta z kluczem black.*.

W drugim eksperymencie czwarty konsument ma klucz black.#, co powinno spowodować przejście do niego wszystkich wiadomości z przedrostkiem black. Wysłałem przykładowe wiadomości (wraz z kluczami):

- message (klucz black.cat) - otrzymali ją konsumenci pierwszy (klucz black.cat), trzeci (klucz black.*) oraz czwarty (klucz black.#)

- nowa wiadomosc (klucz black.) - otrzymali ją konsumenci trzeci (klucz black.*) oraz czwarty (black.#). O ile spodziewałem się, że czwarty konsument dostanie tę wiadomość, tak otrzymanie wiadomości przez konsumenta 3 było dla mnie zaskoczeniem. Możliwe jednak, że * oznacza także słowo puste, co wyjaśniałoby to zdarzenie.
- something (klucz black.cat.something) - tę wiadomość otrzymał już tylko konsument 4 (klucz black.#)
- wiadomosc (klucz blue.cat) - wiadomość, którą otrzymuje już tylko konsument 2 (z kluczem blue.cat)