RabbitMQ Message Broker Desacoplando Workflows – Parte 1

por MundiPagg Blog | abr 26, 2016 | Devs | 0 Comentários

Introdução e Motivação

Com a necessidade cada vez maior das equipes de desenvolvimento responderem rápido a mudanças nos sistemas para representarem novas regras de negócio e a grandes volumes de transações, a arquitetura baseada em serviços de barramento ganha cada vez mais espaço por ser flexível, escalável e segura.

Para solucionar este problema podemos fazer uso do serviço de fila RabbitMQ. Este serviço de fila é open-source, multiplataforma (Windows, Linux, MAC), desenvolvido com Erlang, possui interface de administração Nativa.

Topologia

As topologias básicas do serviço de fila consistem em:

- **Simple:**
  - **Publisher**: agente responsável por incluir uma nova mensagem na fila
  - **MessageBroker (queue)**: serviço de mensageria responsável por manter a informação na fila até que a informação seja consumida.
  - **Consumer**: agente responsável por consumir, retirar, a informação da fila.

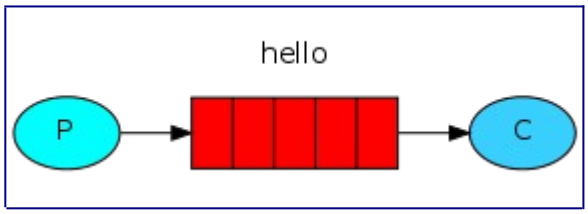


Figura 1 – Fila simples

- **Worker:**

É possível adicionar mais de um **consumer** para dar mais poder de processamento a sua aplicação, processamento em paralelo. Este arranjo é denominado **Worker**.

No RabbitMQ a fila é atômica, ou seja, apenas um item da fila é obtido por vez, não há riscos de mais de um **consumer** obter o mesmo registro. Ele possui internamente um algoritmo de Round-Robin, sendo assim, não é necessário implementar este recurso na implementação do **consumer**.

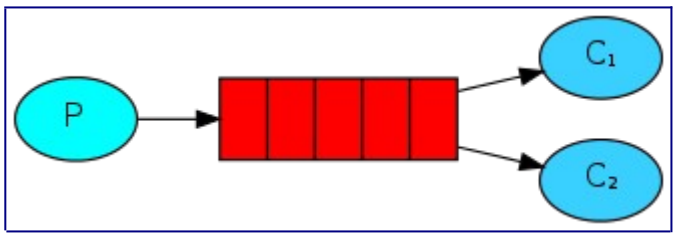
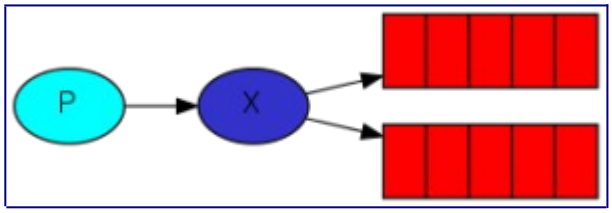


Figura 2 – Fila com mais de um consumer

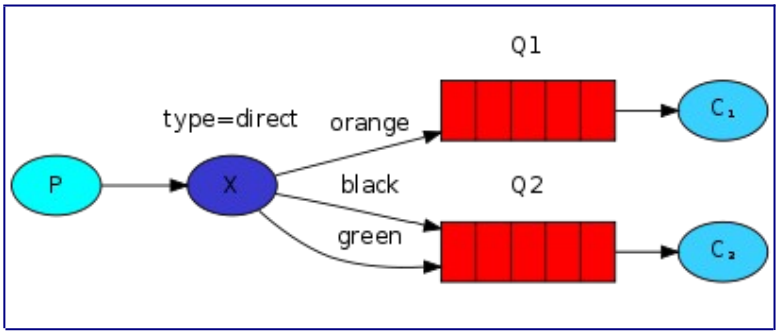
É possível dar mais flexibilidade e controle do fluxo de informação com o uso do Exchange que é responsável por distribuir e/ou filtrar a informação para mais de uma fila com um mínimo de configuração.

Há três topologias que fazem o uso do Exchange:

- **Publish/Subscriber:**
  - Faz a cópia dos dados para mais de uma fila
  - O tipo do Exchange é o **fanout** (esta opção ativa as cópias de mensagens)

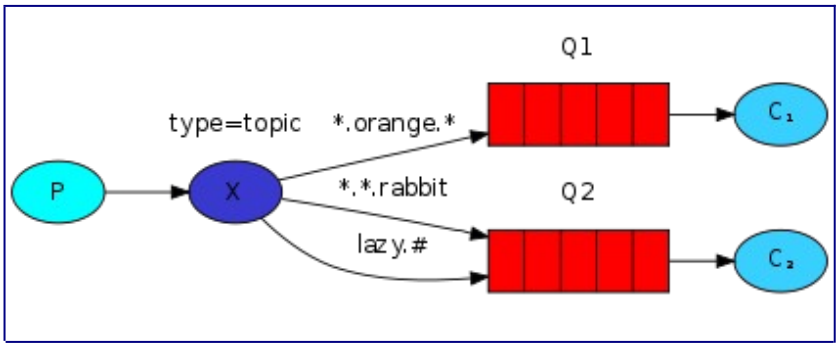


- **Routing**:
  - Analisa a informação do campo routing e direciona a informação para a fila correta.
  - O tipo do Exchange é o **direct** (mensagem direta).
  - Utilize o campo **routing_key** para definir o fluxo da informação.
    - O Publisher e o **cosumer** deve fazer uso do campo routing_key
    - O **consumer** informa o **routing_key** e filtra a mensagem na fila



  - **Topics**:
    - Analisa a informação do campo routing e copia a informação para a fila correta.
    - O tipo de Exchange é o **topic** (permite filtro conforme descrito abaixo).
    - Utiliza a seguinte dois caracteres coringas para efetuar o filtro:
      - * - substitui por um bloco de rota

- # - substitui por um ou mais blocos de rota



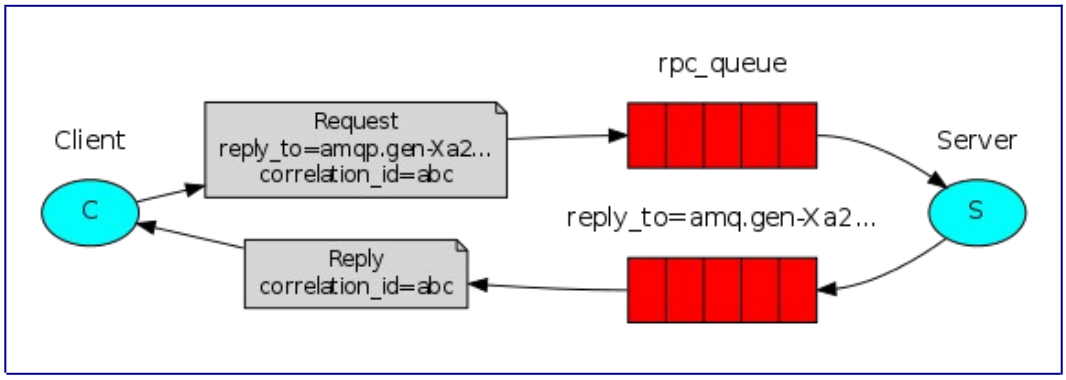
Exemplos:

Se uma mensagem for publicada com o **routing_key:** lazy.rules.test a mensagem é copiada para a fila Q2 e consumido por C2.

Se uma mensagem for publicada com o **routing_key**: lazy.orange.color a mensagem é copiada para a fila Q2 e Q1 e consumido por C1 e C2.

- **RPC:**

Quando há a necessidade de desacoplamento de Workflow, porém, é necessário aguardar a resposta do processamento é recomento o uso da topologia **RPC (Remote Procedure Call)**.



Para uso da topologia RPC, faz-se uso de duas filas, uma para requisição e outra para aguardar a resposta(Callback). Neste tipo de topologia é necessário fazer uso dos seguintes campos:

- **ReplyTo**: nome fila de resposta (Callback).
- **CorrelationId**: identificador de correlação para o **consumer** validar se a informação recebida foi gerada pela mesma requisição efetuada pelo **Publisher**. Server como chave de validação para garantia da informação.

Recursos adicionais do RabbitMQ

O serviço de fila funciona basicamente como uma pilha FIFO (Firt In First Out), porém, há recursos adicionais que tornam o serviço muito mais robusto e flexível. Seguem as principais opções do serviço de fila:

Confirmação de recebimento da mensagem

Para garantir a resiliência na implementação do serviço de fila, o RabbitMQ disponibiliza uma opção de confirmação de mensagem:

- **Ack (acknolodge)**: confirmação de recebimento/processamento.
- **Nack (not acklodge)**: recebimento/processamento não confirmado.

Se a opção de confirmação de recebimento (**Ack**) estiver ativa na criação do **consumer**, após o recebimento/processamento da mensagem, o consumer deve enviar um ACK.

Se houver falha no processamento o **consumer** deve enviar um **Nack** para que a mensagem retorne automaticamente para a fila.

Se o canal de comunicação com o RabbitMQ cair. Ou seja, o **consumer** abortar e desconectar o item também volta automaticamente para a fila para garantir que a mensagem não se perca e possa ser processada pelo próximo **consumer** em atividade.

TTL – Time to Live

O serviço de fila RabbitMQ disponibiliza um recurso interessante chamado **TTL** que define um tempo de vida para informação permanecer na fila, após a expiração, a mensagem fica indisponível e é eliminada da fila automaticamente.

Persistência

Como garantia de entrega das mensagens em caso de indisponibilidade do serviço de fila, é possível configurar as filas como persistente com uso da opção **durable**.

- Persistente: **durable** como **true** (persiste em disco)
- Transiente: **durable** como **false** (somente em memória).

Conclusão

O desacoplamento do Workflow para sistemas legados pode ser feito de maneira gradual, sem grandes impactos na disponibilidade de serviços, desde que, os domínios e a separação por responsabilidade sejam avaliados.

O RabbitMQ é um produto muito poderoso e flexível. Neste artigo foram apresentadas as principais características de funcionamento as opções mais utilizadas. Nos próximos artigos irei apresentar exemplos de utilização e boas práticas e configurações de cluster/replicação.