Materi RabbitMQ

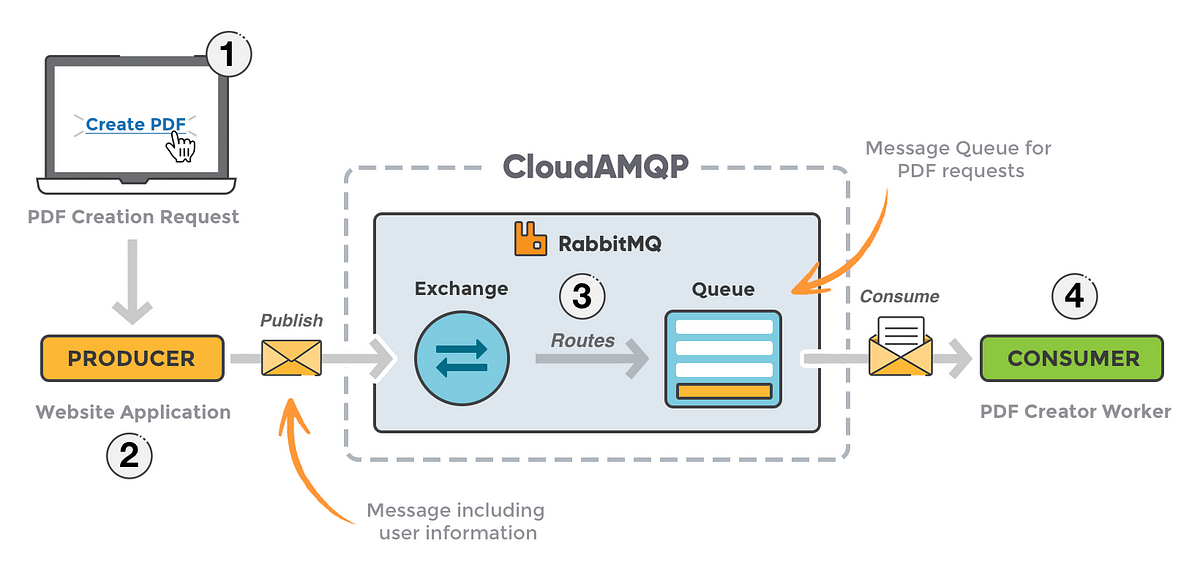
## Event Driven Architecture

Event Driven Architecture (EDA) adalah paradigma arsitektur pengembangan aplikasi yang dimana setiap komponen berkomunikasi melalui peristiwa(event) yang terjadi di dalam sistem. Di dalam arsitektur ini setiap komponen ataupun layanan dikembangkan untuk merespon terhadap setiap event yang terjadi di dalam sistem.

1. **Kelebihan**
2. Loose Coupling
3. Scalability
4. Flexibility and Agility
5. Asynchronous Communication
6. Decoupled Microservices
7. **Kekurangan**
8. Complexity
9. Consistency and Ordering
10. Debugging and Tracing
11. Event Overhead
12. Event Storming and Design Complexity
13. Potential for Event Loss

## Pub/Sub

Publish dan Subscribe di RabbitMQ adalah pola komunikasi di mana produsen (publisher) mengirimkan pesan ke exchange, dan konsumen (subscriber) berlangganan (subscribe) ke exchange tersebut untuk menerima pesan-pesan yang dikirim oleh produsen.



Cara kerja RabbitMQ sebagai berikut:

* 1. Publisher (Penerbit): Penerbit adalah entitas atau komponen yang mengirim pesan ke RabbitMQ. Penerbit mengirim pesan ke pertukaran (exchange) di RabbitMQ.
  2. Exchange (Pertukaran): Pertukaran bertindak sebagai perantara antara penerbit dan pelanggan. Ketika penerbit mengirim pesan ke pertukaran, pertukaran akan memutuskan kemana pesan harus dikirimkan berdasarkan aturan routing tertentu.
  3. Queue (Antrian): Setiap pelanggan memiliki antrian sendiri yang terhubung ke pertukaran. Pesan yang dikirim oleh penerbit ke pertukaran akan didistribusikan oleh pertukaran ke antrian pelanggan yang berlangganan pesan tersebut.
  4. Subscriber (Pelanggan): Pelanggan adalah komponen yang berlangganan ke antrean tertentu di RabbitMQ. Mereka akan menerima pesan yang dikirimkan ke antrian yang mereka langgani.

## Tipe Exchange, Bidding dan Routing

Exchange adalah entitas yang bertanggung jawab untuk menerima pesan dari penerbit (publisher) dan memutuskan cara untuk mendistribusikannya ke antrian (queue).

Binding/routing dalam RabbitMQ adalah hubungan antara exchange dan queue. Binding menentukan queue mana yang akan menerima pesan yang dikirim ke exchange dengan routing key tertentu.

RabbitMQ mendukung beberapa jenis exchange yang memungkinkan Anda mengontrol bagaimana pesan diarahkan ke antrian. Berikut adalah beberapa tipe exchange utama:

* 1. Direct Exchange, mengirimkan pesan ke queue yang terikat dengannya dan memiliki routing key yang sama dengan routing key pesan.
  2. Fanout Exchange, mengirimkan pesan ke semua queue yang terikat dengannya, tanpa memperhatikan routing key.
  3. Topic Exchange, mengirimkan pesan ke queue yang terikat dengannya dan memiliki routing key yang cocok dengan routing key pesan, menggunakan wildcards.
  4. Headers Exchange, mengirimkan pesan ke queue yang terikat dengannya dan memiliki header yang cocok dengan header pesan.
  5. Default Exchange (Nameless Exchange): Default exchange tidak memiliki nama dan diidentifikasi oleh string kosong (""). Routing pada default exchange dilakukan berdasarkan nama antrian. Penerbit dapat langsung mengirim pesan ke antrian dengan menggunakan nama antrian sebagai routing key.

## Acknowledge dan Durable

Acknowledge dan durable adalah dua konsep yang terkait dengan keandalan dan keberlanjutan (durability) dalam sistem pesan RabbitMQ.

* 1. Acknowledge (Ack).
     1. **Acknowledge** adalah konsep di mana setelah seorang konsumen (consumer) menerima dan memproses pesan dari antrian, konsumen memberi tahu RabbitMQ bahwa pesan tersebut telah berhasil diproses.
     2. RabbitMQ menggunakan konsep ack untuk mengonfirmasi bahwa pesan telah diterima dan diproses dengan sukses oleh konsumen.
     3. Jika ack tidak diberikan oleh konsumen (misalnya, karena terjadi kesalahan dalam pemrosesan pesan), RabbitMQ dapat menganggap pesan tersebut belum diproses dan mengirimkannya ke konsumen lain atau menyimpannya kembali di antrian.
     4. Acknowledge membantu dalam mencapai ketersediaan dan keandalan sistem pesan.
  2. Durable.
     1. **Durable** mengacu pada sifat persisten atau tahan lama dari antrean dan pertukaran RabbitMQ.
     2. Jika antrean atau pertukaran dinyatakan sebagai durable, mereka akan tetap ada bahkan setelah server RabbitMQ restart. Hal ini memastikan bahwa konfigurasi dan data tidak hilang.
     3. Pesan juga dapat diberi atribut durability. Jika pesan dinyatakan sebagai durable, RabbitMQ akan menyimpannya ke disk, sehingga pesan tersebut tidak akan hilang meskipun RabbitMQ restart.
     4. Durability membantu dalam menjaga konsistensi dan keberlanjutan sistem pesan.

## Prefetch

**Prefetch** dalam konteks RabbitMQ adalah cara untuk mengontrol seberapa banyak pesan yang dapat dikonsumsi oleh konsumen (consumer) sebelum konsumen memberikan acknowledgment (ack) kepada RabbitMQ. Penggunaan prefetch dapat membantu dalam mengatur throughput dan menghindari overconsumption, terutama ketika ada banyak pesan dalam antrian.

Dengan menggunakan prefetch, Anda dapat mengontrol berapa banyak pesan yang dikirimkan kepada konsumen sebelum konsumen memberikan ack untuk pesan-pesan tersebut. Ini dapat berguna untuk mencegah konsumen mengkonsumsi terlalu banyak pesan sekaligus, yang dapat menyebabkan kinerja yang buruk atau bahkan kehabisan sumber daya.

## Channel dan Connection

Dalam konteks RabbitMQ, terdapat dua konsep penting yang berkaitan dengan koneksi dan saluran komunikasi:

* 1. Connection (Koneksi):
     1. Koneksi (connection) adalah koneksi fisik antara aplikasi atau komponen perangkat lunak dengan server RabbitMQ.
     2. Setiap koneksi mengidentifikasi sesi terpisah antara aplikasi dan server RabbitMQ. Koneksi ini biasanya melibatkan pertukaran informasi autentikasi dan pengaturan awal lainnya.
     3. Koneksi memungkinkan aplikasi berkomunikasi dengan broker RabbitMQ dan mentransmisikan pesan
  2. Channel (Saluran):
     1. Saluran (channel) adalah saluran virtual di dalam koneksi fisik yang dapat digunakan untuk mentransmisikan pesan.
     2. Satu koneksi dapat memiliki banyak saluran. Setiap saluran beroperasi secara terisolasi di dalam koneksi, memungkinkan aplikasi mengelola beberapa saluran tanpa harus membuat koneksi tambahan.
     3. Saluran memungkinkan paralelisme dalam komunikasi, dan penggunaannya lebih efisien daripada membuka koneksi baru untuk setiap interaksi.

## Retry dan Health Check

Retry dan health check adalah dua konsep penting dalam pengelolaan sistem dan kesehatan (health) RabbitMQ, terutama dalam skenario di mana ada kebutuhan untuk mengatasi kesalahan komunikasi atau memastikan bahwa sistem beroperasi dengan baik. Berikut adalah cara-cara umum untuk menerapkan retry dan health check pada RabbitMQ:

### Retry:

* 1. Exponential Backoff:
     1. Implementasikan skema waktu tunggu eksponensial untuk melakukan retry. Artinya, waktu tunggu antara setiap upaya retry bertambah secara eksponensial.
     2. Ini membantu mencegah oversaturation pada saat kesalahan dan memberikan waktu untuk pemulihan sistem.
  2. Dead Letter Exchange (DLX):
     1. Konfigurasikan antrian dengan Dead Letter Exchange. Jika pesan tidak dapat diproses setelah sejumlah percobaan tertentu, kirim pesan ke Dead Letter Exchange.
     2. Pesan yang dikirim ke Dead Letter Exchange dapat diinspeksi atau diolah lebih lanjut.
  3. Retry Counter:
     1. Tambahkan header khusus ke pesan untuk melacak jumlah upaya retry yang telah dilakukan.
     2. Pada setiap percobaan, konsumen dapat memeriksa header tersebut dan memutuskan apakah akan melakukan retry atau mengirim pesan ke Dead Letter Exchange.

### Health Check:

1. Monitoring dan Alerting
   1. Gunakan alat pemantauan untuk memonitor kesehatan RabbitMQ, termasuk statistik performa, status koneksi, dan status antrian.
   2. Atur notifikasi atau alerting untuk memberi tahu ketika ada masalah atau ketidaknormalan.
2. HTTP Health Check Endpoint:
   1. Implementasikan endpoint HTTP khusus pada aplikasi Anda yang berinteraksi dengan RabbitMQ untuk menyediakan informasi status kesehatan.
   2. Endpoint ini dapat memberikan respons berdasarkan status RabbitMQ, jumlah pesan di antrean, atau parameter kesehatan lainnya.
3. Connection Heartbeats
   1. Aktifkan heartbeats pada level koneksi. RabbitMQ dapat mengirim heartbeats secara berkala dan mengharapkan balasan dari klien.
   2. Jika koneksi terputus atau tidak menerima heartbeats, RabbitMQ dapat menganggap koneksi tersebut tidak sehat.
4. Queue Length Monitoring
   1. Memantau panjang antrian secara teratur. Jika jumlah pesan di antrian melebihi batas tertentu, ini dapat menjadi indikator bahwa sistem tidak dapat menangani beban pekerjaan saat ini.