# RabbitMQ

## Messaging broker kod mikroservisnih aplikacija

**Marko Igic 1936**

# Sadrzaj

- Sta je RabbitMQ?
- Osnovni pojmovi kod RabbitMQ-a
- Kada izabrati RabbitMQ
- Slozen model rutiranja - flow slanja poruka
- Prednosti i mane RabbitMQ-a
- Konkurentna resenja
- Projekti

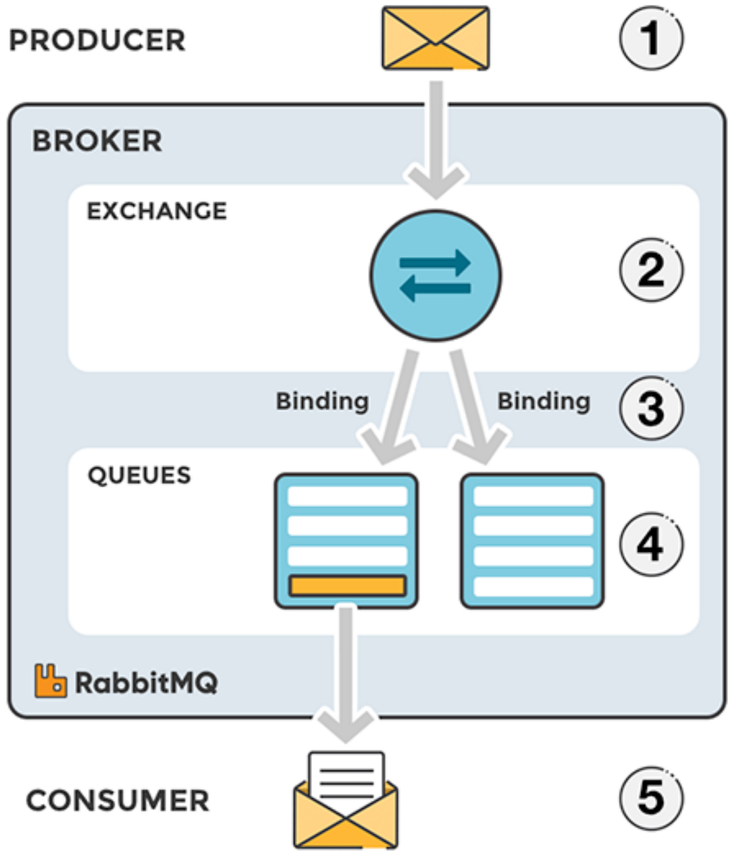
# Sta je RabbitMQ ?

- Sistem za slozenu razmenu poruka (messaging broker)
- Fleksibilan nacin razmene poruka

  1. Point-to-point(1 producer, 1 consumer)

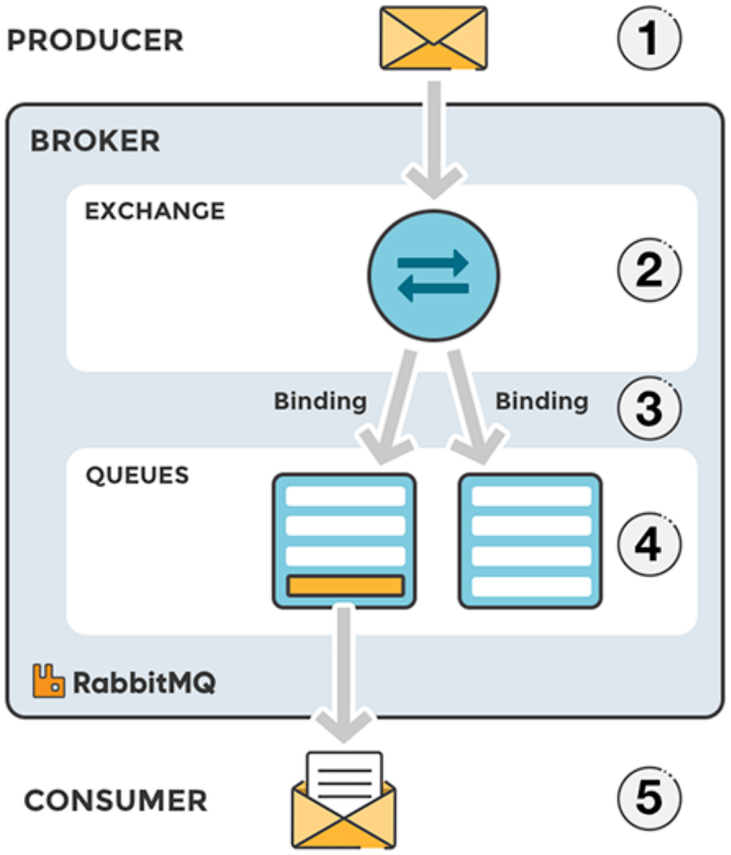
  2. Publish - subscribe (1 producer, n consumer-a)

- Baziran na AMQP (Advanced Message Queue Protocol)
- Razliciti tipovi poruka



# Osnovni pojmovi kod RabbitMQ-a

1. **Producer** - mikroservis/aplikacija koja salje poruku
2. **Queue** - veliki buffer poruka (struktura podataka FIFO), cija je memorija ogranicena memorijom host-a. Moze imati veci broj producer-a i consumer-a
3. **Consumer** - mikroservis/aplikacija koja je zaduzena za prihvatanje i obradu poruke
4. **Exchange** - “manager” koji prosledjuje poruke od producer-a do queue-eva u zavisnosti od definisanih pravila
5. **Binding** - veza izmedju Queue-a i Exchange-a
6. **Routing key** - kljuc koji sugerise Exchange-u na koji nacin tj. Dokle treba rutirati poruku



# Kada izabrati RabbitMQ?

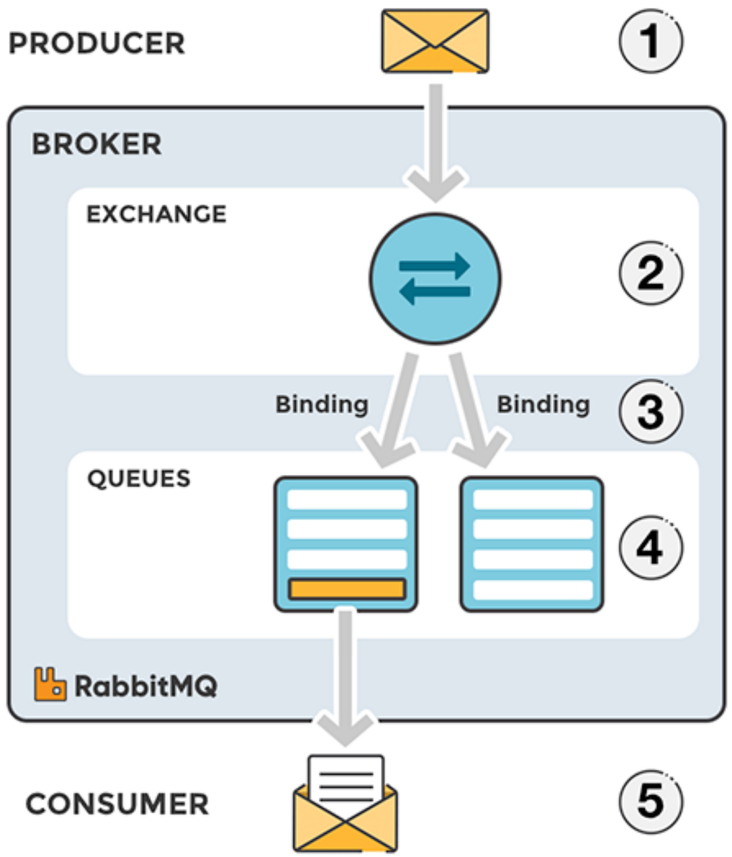
1. Slozeno rutiranje
2. Prioretizacija poruka - prioritetni Queue-evi
3. Obrada dugotrajnih zahteva
4.Pouzdanost
5. Jednostavna integracija sa razlicitim tehnologijama - Java, Python, .NET, Swift, Go …

# Slozen model rutiranja poruka

***!!! Nije preporucljivo poruke slati bez uticaja Exchange-a***

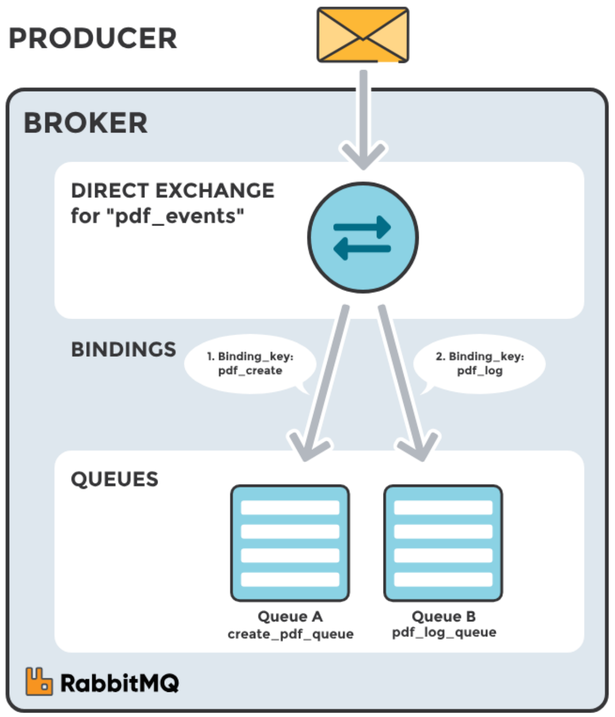
## Flow slanja poruka:

1. Producer salje poruku do Exchange-a. Kada se pravi Exchange mora da se definise njegov tip.
2. Exchange je prihvatio poruku i sada je zaduzen za njeno rutiranje do odgovarajuceg Queue-a.
3. Mora da se napravi veza izmedju Exchange-a i Queue-a.
4. Poruke ostaju u Queue-u sve dok ne budu handle-ovane
5. Consumer preuzima i obradjuje poruke

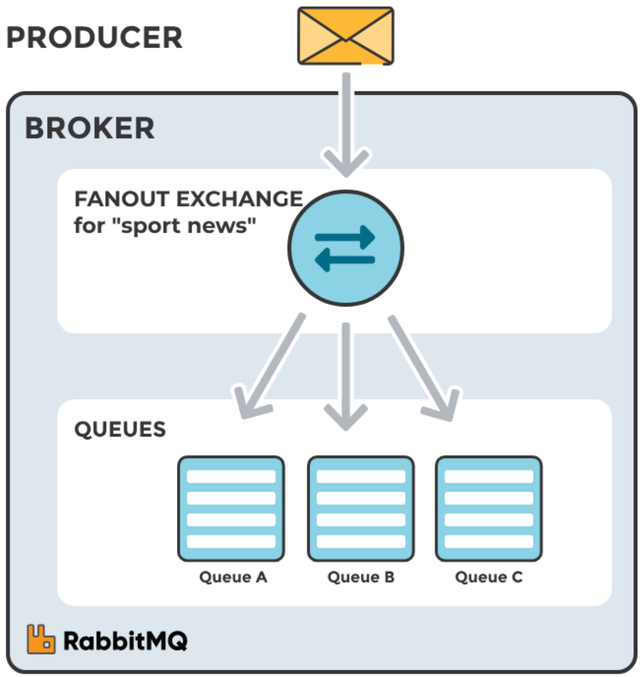


# Tipovi rutiranja (exchange-a)

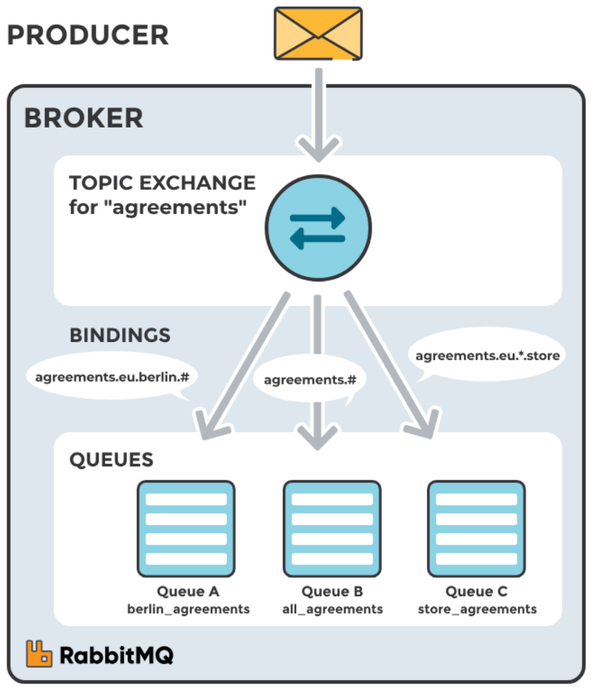
## DIRECT EXCHANGE




## FANOUT EXCHANGE




## TOPIC EXCHANGE



# Prednosti i mane

- Centralizovan sistem, koji upravlja porukama na jednom mestu
- Koristi mehnizam potvrde poruka
- Fleksibilna mogucnost rutiranja poruka
- Open source project
- Veliki obim poruka (do 1 milion u sekundi)
- Podrska za autentifikaciju

- Nije optimizovan za velike kolicne podataka (BigData)
- Uprkos open-source-u, dodatni troskovi
- Problem centralizovanog sistema
- Ogranicena horizontally skalabilnost

# Paralela sa konkurentnim resenjima

## Konkurenti: Apache Kafka i ActiveMQ

| | Apache Kafka | RabbitMQ | ActiveMQ |
|---|---|---|---|
| **Performanse i skalabilnost** | 1. Velika propusnost<br>2. Horizontalna skalabilnost | 1. Dobre performanse za manje projekte<br>2. Losije u odnosu na Kafku | 1. Dobre performanse za manje projekte<br>2. Losije u odnosu na Kafku |
| **Prioretizacija poruka** |  | Prioritetni queue-evi | Podrzano |
| **Trajnost poruka** |  |  |  |
| **Rutiranje poruka** | Podrzano na osnovu partiocionisanja u okviru Topic-a | Podrzano na osnovu Exchange-a i bindinga | Podrzano na pomocu selektora i Topic-a |
| **Replikacija** | Ugradjena (replication factor) | Mirrored Queues, Quorum Queues, Federated Queues | Master-slave replikacija |

# Jednostavna implementacija - Round Robin

RR RoundRobin  Version control  Current File

Producer.py

```python
import pika
import sys
import os

# First argument is message that is set to be sent
def send_message():  1 usage
    message = sys.argv[1]

    connection = pika.BlockingConnection(
        pika.ConnectionParameters(host="localhost")
    )
    channel = connection.channel()
    channel.queue_declare(queue="Round-Robin")

    channel.basic_publish(
        exchange="",
        routing_key="Round-Robin",
        body=message
    )
    print(f"Producer sent : {message}")

    channel.close()

if __name__ == "__main__":
    try:
        send_message()
```

Consumer.py

```python
import os
import sys
import pika

def main():  1 usage
    connection = pika.BlockingConnection(
        pika.ConnectionParameters(host="localhost"),
    )
    channel = connection.channel()

    channel.queue_declare(queue="Round-Robin")

    def callback(ch, method, properties, body):
        print(f"Received : {body.decode()}")

    channel.basic_consume(
        queue="Round-Robin",
        on_message_callback=callback,
        auto_ack=True,
    )

    channel.start_consuming()

if __name__ == "__main__":
    try:
        main()
```

# Work Queues - nadogradnja

- Raspodela poruka tek po zavrsenoj obradi poruke
- Dodato je potvrdjivanje poruka
- Acknowledgment poruka se salje na isti queue !
- Pouzdanost poruka - slucajevi otkaza

```
channel.queue_declare(queue='task_queue', durable=True)
```

**- Trajan Queue**

```
channel.basic_publish(exchange='',
                      routing_key="task_queue",
                      body=message,
                      properties=pika.BasicProperties(
                         delivery_mode = pika.DeliveryMode.Persistent
                      ))
```

**- Trajne poruke koje se salju na Queue**

Producer.py

```python
import pika
import sys
import os

def send_message():  1 usage
    message = ''.join(sys.argv[1:])
    print(message)

    connection = pika.BlockingConnection(
        pika.ConnectionParameters(host="localhost")
    )
    channel = connection.channel()
    channel.queue_declare(queue="workers")
    channel.basic_publish(
        exchange='',
        routing_key="workers",
        body=message,
    )
    connection.close()


if __name__ == "__main__":
    try:
        send_message()
    except KeyboardInterrupt:
        try:
```

Consumer.py

```python
import time


def main():  1 usage
    connection = pika.BlockingConnection(
        pika.ConnectionParameters(host="localhost")
    )
    channel = connection.channel()
    channel.queue_declare(queue="workers")

    def callbackFunc(ex,method, properties, body):
        print(f"Received : {body.decode()}")
        time.sleep(body.count(b'.')) #Counts number of ".", and
        channel.basic_ack(delivery_tag=method.delivery_tag) # Th

        #If auto_ack is set, message won't be deleted if an erro

    channel.basic_qos(prefetch_count=1) # Consumer receives mess
    channel.basic_consume(
        queue="workers",
        on_message_callback=callbackFunc)

    channel.start_consuming()

if __name__ == "__main__":
```

# Publish/Subscribe - fanout Exchange

Publisher.py

```python
import pika
import sys
import os

#Application sends message to all Subscribers (Fanout Exchange)
def send_message():  1 usage
    message = input("Input message to send all Subscribers - ")

    connection = pika.BlockingConnection(
        pika.ConnectionParameters(host="localhost")
    )
    channel = connection.channel()
    channel.exchange_declare(exchange="manager",exchange_type="fanout")

    channel.basic_publish(
        exchange="manager",
        routing_key="",
        body=message
    )
    print(f"Mesage sent : {message}")
    connection.close()

if __name__ == "__main__":
    try:
        send_message()
    except KeyboardInterrupt:
        try:
            sys.exit(0)
        except SystemExit:
            os._exit(0)
```

Subscriber.py

```python
# Subscriber can receive messages ONLY after it has been st
def main():  1 usage
    print("Starts listening")

    connection = pika.BlockingConnection(
        pika.ConnectionParameters(host="localhost")
    )
    channel = connection.channel()
    channel.exchange_declare(
        exchange="manager",
        exchange_type="fanout"
    )
    result = channel.queue_declare(
        queue="",
        exclusive=True #enables queue to be deleted after c
    )
    queue_name = result.method.queue
    channel.queue_bind(
        exchange="manager",
        queue=queue_name,
        routing_key=""
    )
    def callback(ch, method, properties, body):
        print(f"Subscriber received : {body.decode()}")

    channel.basic_consume(
        queue=queue_name,
        on_message_callback=callback,
        auto_ack=True
    )

    channel.start_consuming()
```

# Topic exchange

ublisher.py

```python
import pika
import sys
import os

def send_message():  1 usage
    print("type1.type2.type3.type4 ...")
    routing_key = input("Enter which subscribers the message is intended for (
    message_body = input("Enter message body - ")

    connection = pika.BlockingConnection(
        pika.ConnectionParameters(host="localhost")
    )
    channel = connection.channel()
    channel.exchange_declare(
        exchange="topicExchange",
        exchange_type="topic"
    )
    channel.basic_publish(
        exchange="topicExchange",
        routing_key=routing_key,
        body=message_body
    )

    print(f"{routing_key} : {message_body}")

    connection.close()

if __name__ == "__main__":
    try:
        send_message()
    except KeyboardInterrupt:
        try:
            sys.exit(0)
        except SystemExit:
            os._exit(0)
```

Subscriber.py

```python
def read_message():  1 usage
    print("EXAMPLE : type1.type2.type3.type4 ...")
    print("typeX can be replaced with # which means 0 or more words")
    print("typeX can be replaced with * which means 1 word")
    routing_key = input("Enter routing key in appropriate format - ")

    connection = pika.BlockingConnection(
        pika.ConnectionParameters(host="localhost")
    )
    channel = connection.channel()
    channel.exchange_declare(
        exchange="topicExchange",
        exchange_type="topic"
    )
    result = channel.queue_declare(
        queue = "",
        exclusive= True
    )
    queue_name = result.method.queue
    channel.queue_bind(
        queue=queue_name,
        exchange="topicExchange",
        routing_key=routing_key
    )


    def callback(ch, method, properties, body):
        print(f"{method.routing_key} : {body.decode()}")

    channel.basic_consume(
        queue= queue_name,
        on_message_callback= callback,
        auto_ack= True
    )
    channel.start_consuming()
```

# Topic exchange - demonstracija

```
Topic — -zsh — 80×24
markoigic@Markos-MacBook-Pro Topic % python3 Publisher.py
type1.type2.type3.type4 ...
Enter which subscribers the message is intended for (in appropriate format) type
1.type2.type3 - balkan.serbia.nis
Enter message body - Weather is nice and sunny
balkan.serbia.nis : Weather is nice and sunny
markoigic@Markos-MacBook-Pro Topic %
```

```
Topic — Python Subscriber.py — 80×24
markoigic@Markos-MacBook-Pro Topic % python3 Subscriber.py
EXAMPLE : type1.type2.type3.type4 ...
typeX can be replaced with # which means 0 or more words
typeX can be replaced with * which means 1 word
Enter routing key in appropriate format - balkan.serbia.#
balkan.serbia.nis : Weather is nice and sunny
```

```
Topic — Python Subscriber.py — 80×24
markoigic@Markos-MacBook-Pro Topic % python3 Subscriber.py
EXAMPLE : type1.type2.type3.type4 ...
typeX can be replaced with # which means 0 or more words
typeX can be replaced with * which means 1 word
Enter routing key in appropriate format *.nis
```

## Napomene:

**Reci su odvojene tackom (.)**

**# zamenjuje jednu ili vise reci**

*** zamenjuje tacno jednu rec**