# Практика 7: RabbitMQ

Юрий Литвинов
y.litvinov@spbu.ru

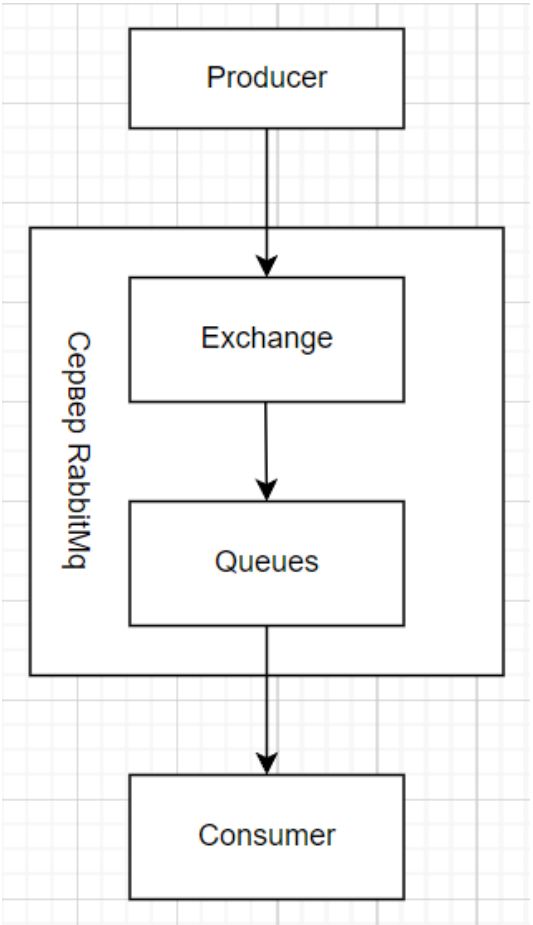
19.05.2022

# RabbitMQ

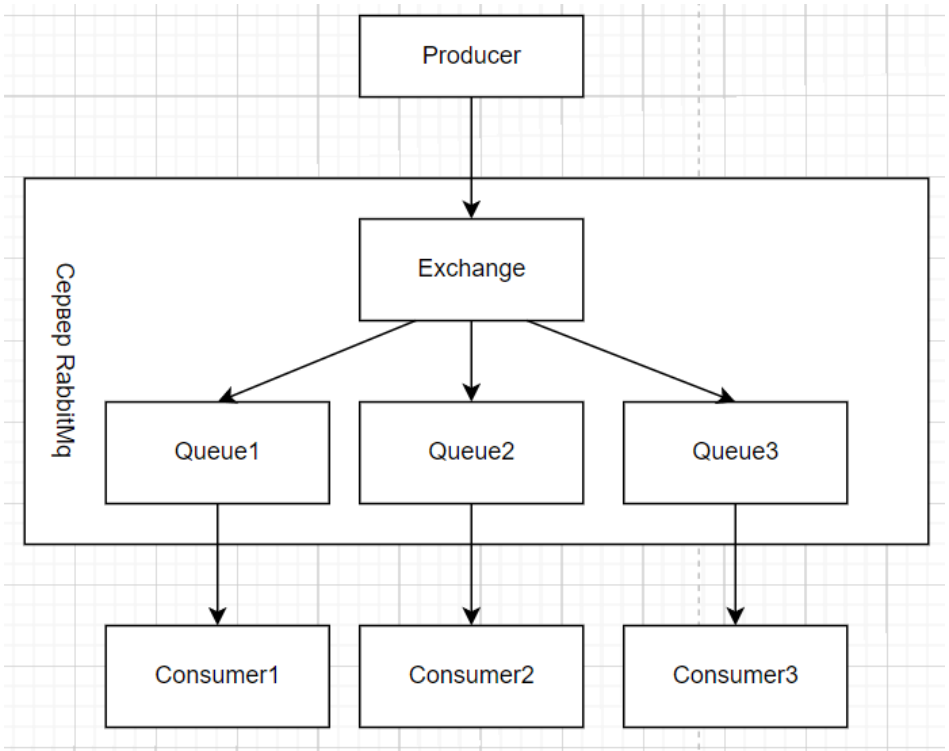
- Сервер и клиенты системы надёжной передачи сообщений
  - Сообщение посылается на сервер и хранится там, пока его не заберут
  - Продвинутые возможности по маршрутизации сообщений
- Реализует протокол AMQP (Advanced Message Queuing Protocol), но может использовать и другие протоколы
- Сервер написан на Erlang, клиентские библиотеки доступны для практически чего угодно

# Архитектура RabbitMQ

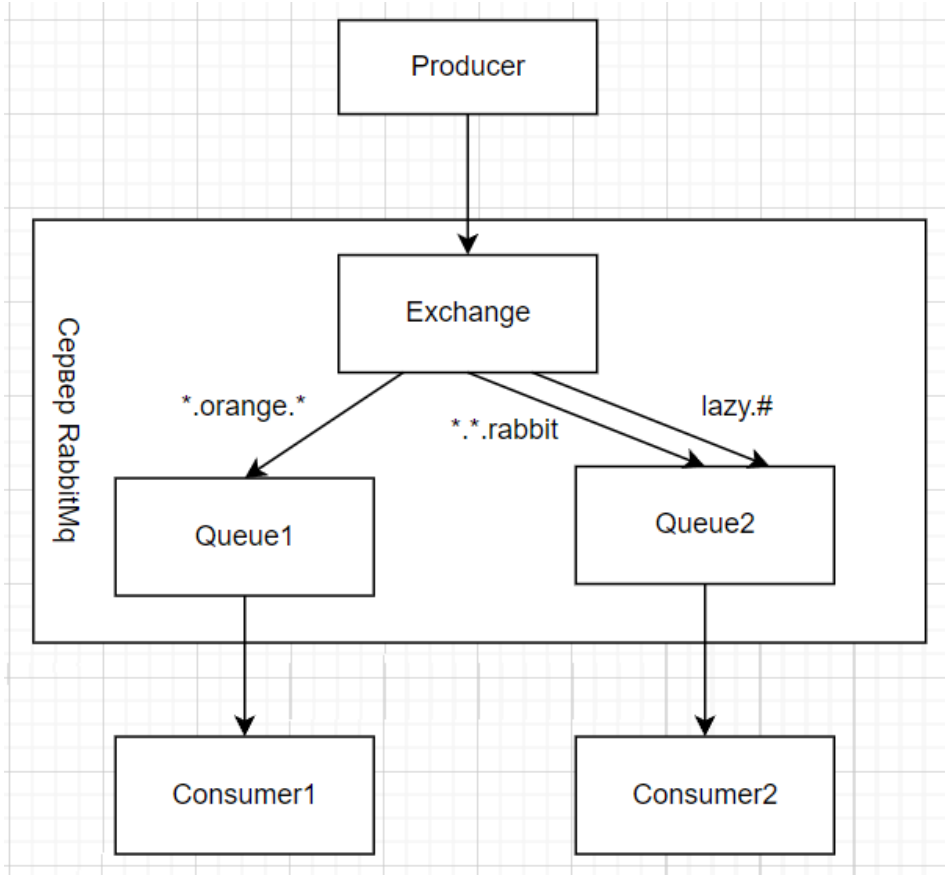
- Producer шлёт сообщения
- Exchange их маршрутизует
- Queue их хранит
- Consumer их забирает



# Пример, Fanout Exchange



# Пример, Direct Exchange с темами




- Routing key — метаинформация для диспетчеризации сообщений (вида «слово.слово.слово...»)

# Пример, отправитель

```
using RabbitMQ.Client;
using System.Text;

var factory = new ConnectionFactory() { HostName = "localhost" };
using var connection = factory.CreateConnection();
using var channel = connection.CreateModel();

channel.QueueDeclare(queue: "hello",
    durable: false,
    exclusive: false,
    autoDelete: false,
    arguments: null);

var message = "Hello World!";
var body = Encoding.UTF8.GetBytes(message);

channel.BasicPublish(exchange: "",
    routingKey: "hello",
    basicProperties: null,
    body: body);

Console.WriteLine($" [x] Sent {message}");
```

# Пример, получатель

```
using RabbitMQ.Client;
using RabbitMQ.Client.Events;
using System.Text;

var factory = new ConnectionFactory() { HostName = "localhost" };
using var connection = factory.CreateConnection();
using var channel = connection.CreateModel();

channel.QueueDeclare(queue: "hello",
    durable: false,
    exclusive: false,
    autoDelete: false,
    arguments: null);

var consumer = new EventingBasicConsumer(channel);

consumer.Received += (model, ea) =>
{
    var body = ea.Body.ToArray();
    var message = Encoding.UTF8.GetString(body);
    Console.WriteLine($" [x] Received {message}");
};

channel.BasicConsume(queue: "hello",
    autoAck: true,
    consumer: consumer);
```

# Как всё собрать и запустить

- Использовать официальный Docker-образ, или:
- Поставить рантайм Erlang
- Поставить сервер RabbitMQ
  - https://www.rabbitmq.com/download.html
- Добавить зависимость от клиента RabbitMQ в проект
  - .NET: RabbitMQ.Client в NuGet
  - JVM:
    compile group: 'com.rabbitmq', name: 'amqp-client', version: '5.14.2'
- Пролистать Getting Started
  - https://www.rabbitmq.com/getstarted.html, часть 1

# Задача на пару

В командах по два человека реализовать консольный сетевой чат на RabbitMQ

- Сервер для обмена сообщениями, о котором договариваются клиенты
  - Центральный сервер, задаваемый как параметр командной строки (127.0.0.1 по умолчанию)
- Есть именованные каналы, на которые можно переключаться и постить туда
  - Должна быть команда подписки на канал, типа «!switch channel1»
  - Начальный канал принимается как аргумент командной строки
  - Переключение на несуществующий канал должно его создавать
- Нет истории, получать только те сообщения, что были опубликованы с момента подключения
  - Может помочь https://www.rabbitmq.com/tutorials/tutorial-three-dotnet.html