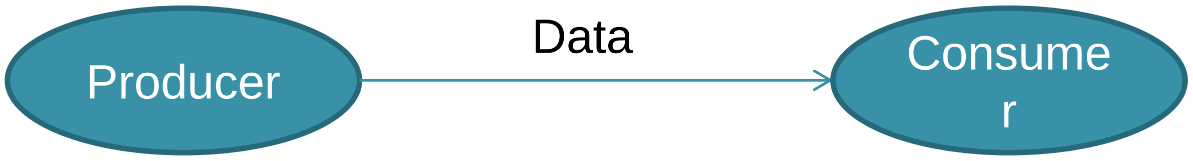
RabbitMQ Introduction

Scope of this presentation

- In Scope
  - Brief history of Rabbit MQ
  - Brief introduction of Rabbit MQ components
  - Messaging Flow in Rabbit MQ
  - Example of sending and receiving messages to/from RabbitMQ

Data Flow

- Data is every where
- Data flows from
  - Method – Method
  - Class – Class
  - Module – Module
  - System – System



Why Messaging systems

- Loose coupling between modules
- Queuing data for later delivery
- Asynchronous processing
- Reliable load balancing

Rabbit MQ

- Rabbit Technologies started as a joint venture between LShift and CohesiveFT in 2007
- acquired in April 2010 by SpringSource, a division of VMWare.
- Became part of GoPivotal in May 2013
- Client Libraries available for all major languages

RabbitMQ Introduction

- **RabbitMQ** is open source Messaging Broker software
- Message Oriented Middleware
- Server written in Erlang
- Implements AMQP

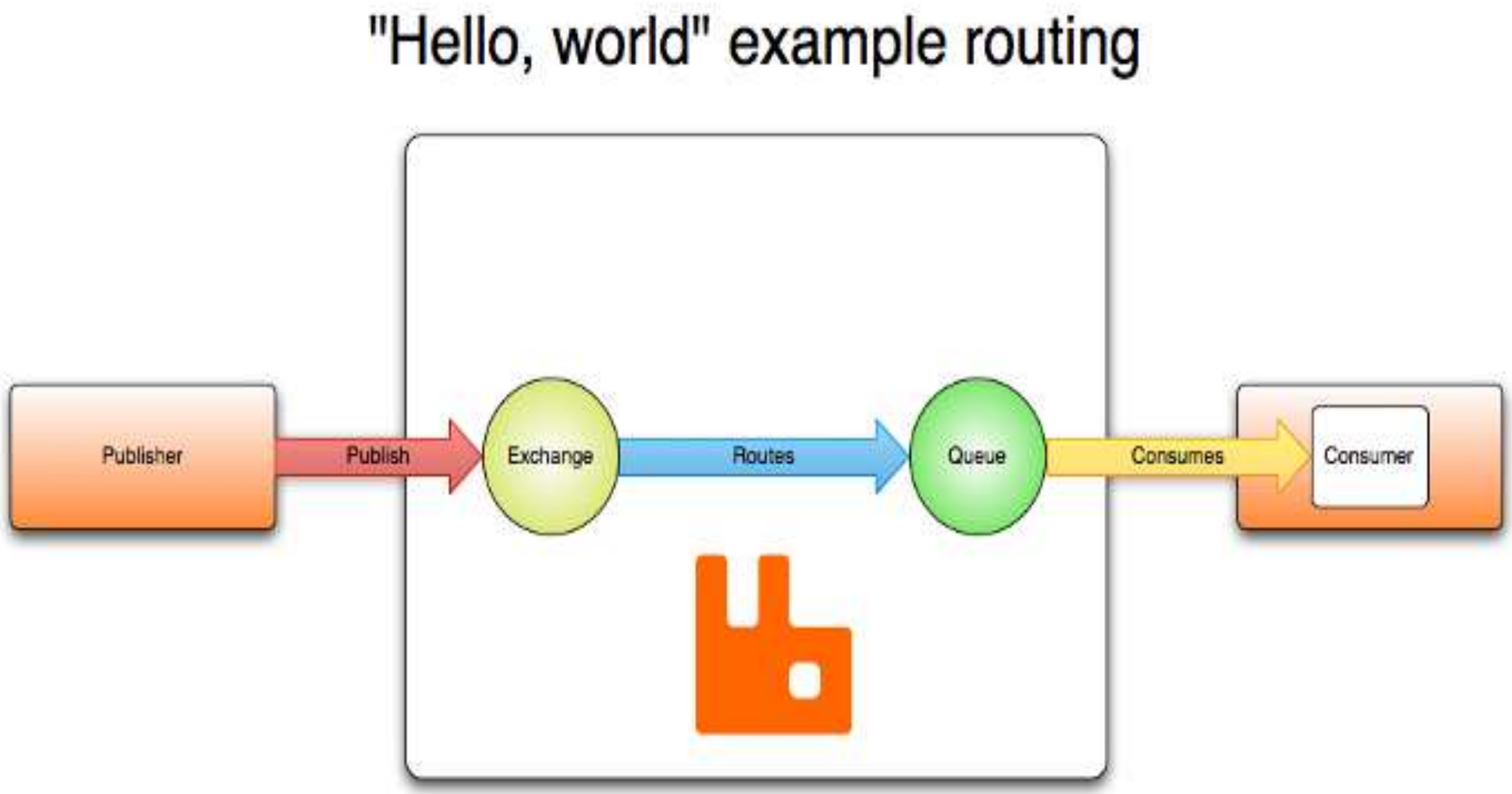
Differences between JMS and AMQP

JMS	AMQP
API	Protocol
5 different data types	Only supports binary Data type
2 messaging models P-P and Publish Subscriber	4 messaging models (Exchanges)
Producers sends messages to Queue/Topic directly	Producers sends messages to an exchanges
Java specific	Has support for many languages Java, Pika, Ruby

Java Client API

```
//Getting Channel
        ConnectionFactory factory = new ConnectionFactory();
        factory.setHost(“<host>");
        Connection connection = factory.newConnection();
        Channel channel = connection.createChannel();
// Declare the exchange
         channel.exchangeDeclare(EXCHANGE_NAME, "direct");
// declare and bind the queue to exchange
          String queueName = channel.queueDeclare(QUEUE_NAME,
false, false, false, null).getQueue();
          channel.queueBind(queueName, EXCHANGE_NAME, "<routing
key>");
//publish message to queue
        String message = "Hello World!";
        channel.basicPublish("", queueName, null,    message.getBytes());
```

Basic Flow


"Hello, world" example routing
Publisher
Publish
Exchange
Routes
Queue
Consumes
Consumer

Binding and Routing Key

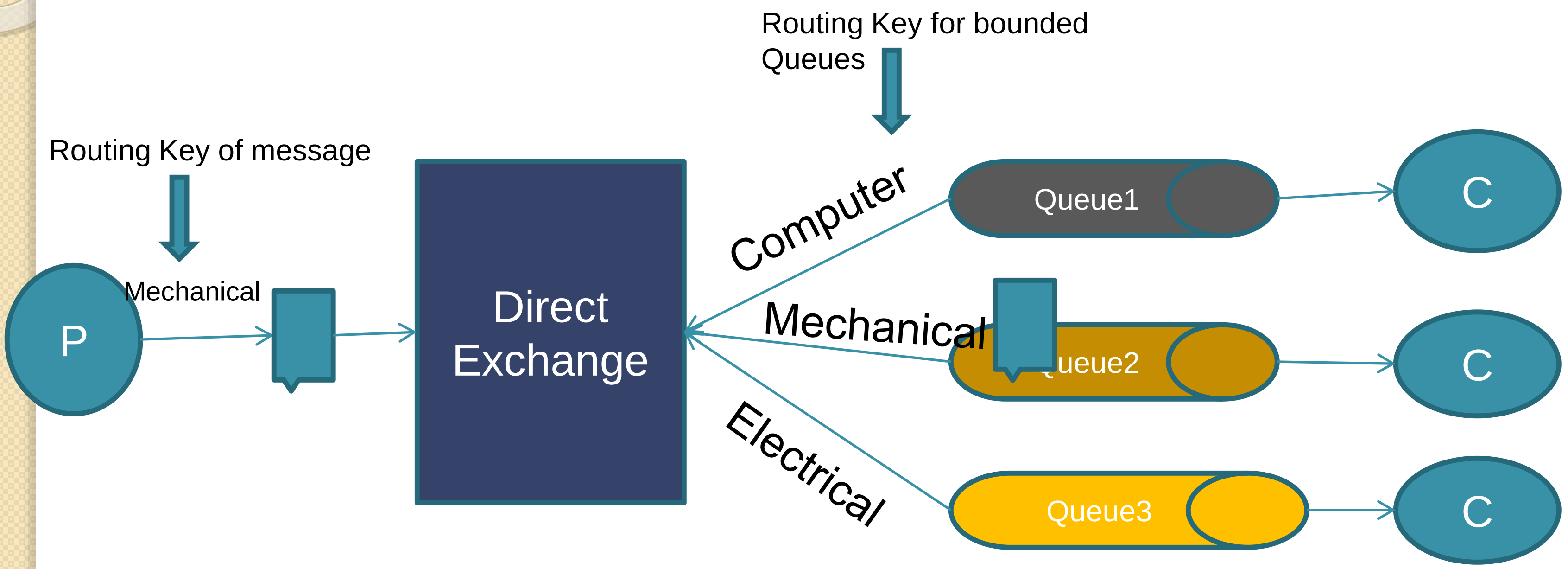
- Bindings: Each Queue should bind with an Exchange with a *routing key*
- Routing key : String of characters
- Exchange Type decides strategy for routing messages to bounded queues
  - Direct exchange : matches routing key of message exactly with the routing key of queues specified at the time of binding
  - Fanout exchange : ignores the routing key and sends a copy of message to every bounded queue

Exchange

- Default
  - Direct exchange with no name
  - Routing key equals QueueName
- Direct
  - Messages would only be routed when there is queue bonded with Routing Key
- Fanout
  - Messages would be routed to all queues bounded irrespective of routing key
- Topic
  - We can use regular expressions for routing key
- Header
  - Least used
  - Matches against header properties instead of routing key

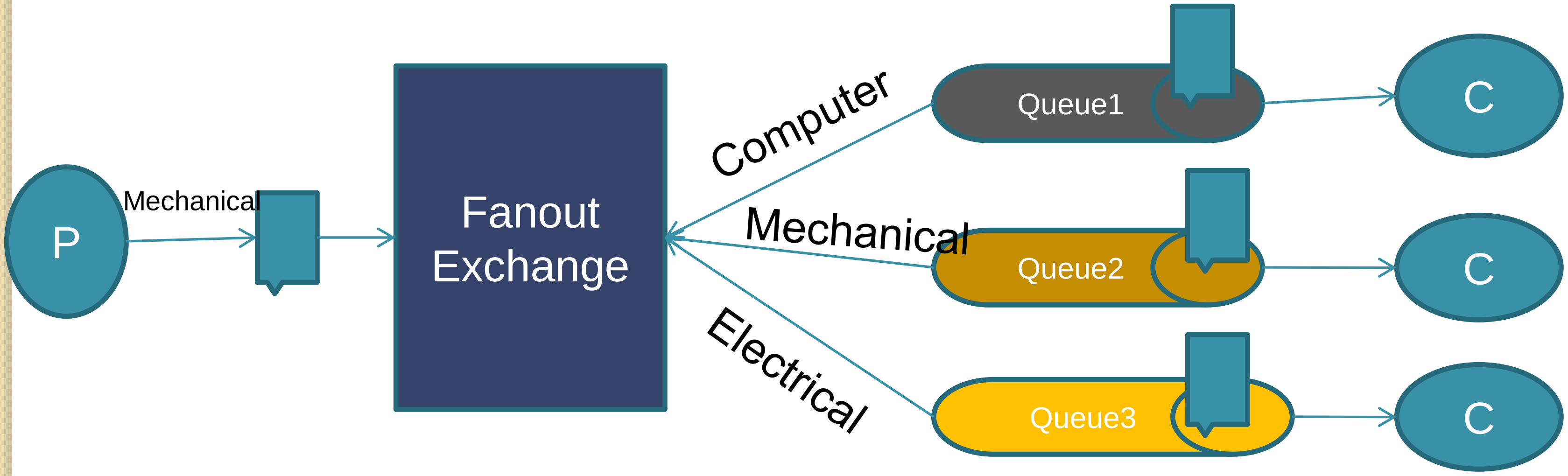
Direct Exchange

(Peer – Peer Communication with RabbitMQ)



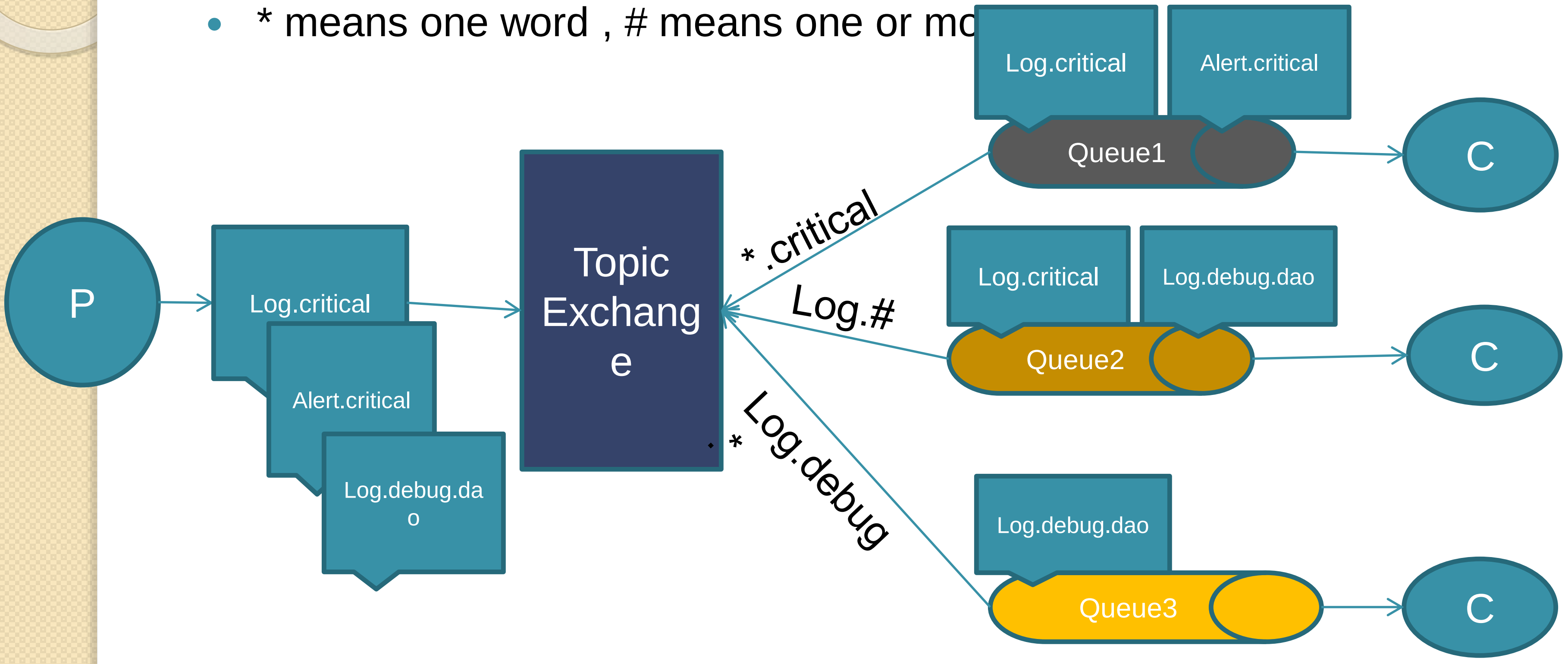
Fanout Exchange

(Publish Subscriber model with RabbitMQ)



Topic exchange

- Regular expressions as routing key
- * means one word , # means one or mo



Code Samples (Git repository)

- https://github.com/ShirishkumarBari/LearnRabbitMQ

References

- https://www.rabbitmq.com/
- https://en.wikipedia.org/wiki/RabbitMQ
- http://www.levvel.io/blog-post/rabbitmq-a
  introduction/
- https://dzone.com/