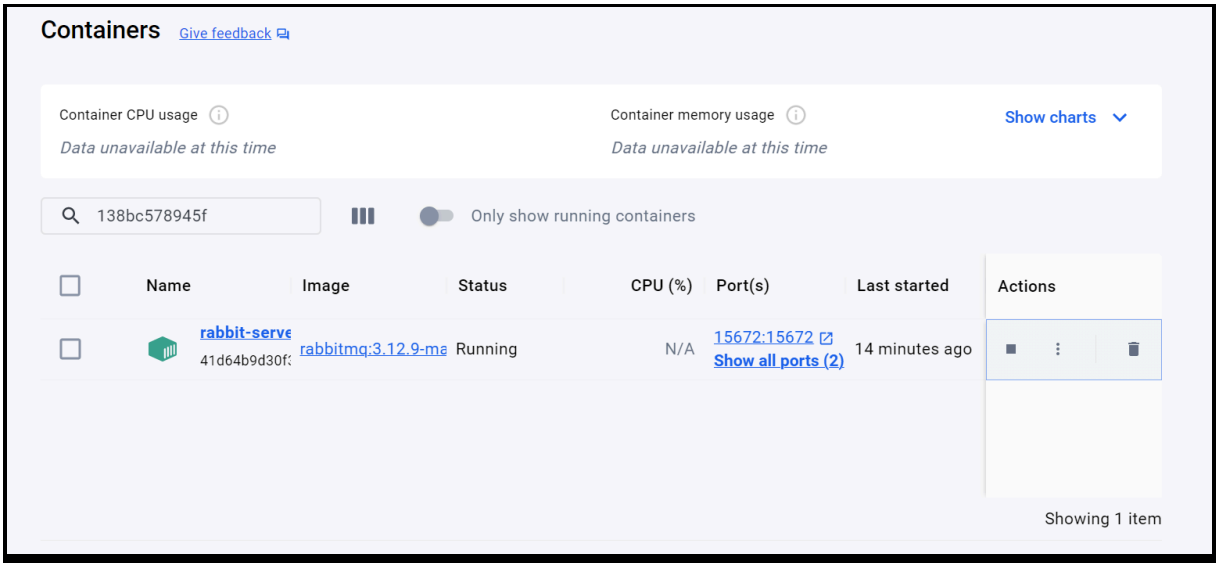
TP: RabbitMQ (Un producteur et deux consummer)
TP réalisé par afsa Karchaou (5iir G7)TP:

Partie 1:

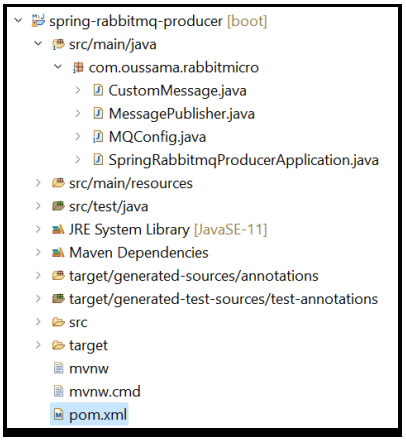
Installation de l'image Docker:

```
docker pull rabbitmq:3.12.9-management
```



Partie 2:

Réalisation du microservice 1 contenant le producteur:



- application.properties

```
server.port = 8123
 spring.rabbitmq.addresses = localhost:5672
```

- Configuration RabbitMQ avec 2 exchanges et 2 queues

```
package com.oussama.rabbitmicro;
```

```
import org.springframework.amqp.core.*;
import
   org.springframework.amqp.rabbit.connection.ConnectionFactory;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import
   org.springframework.amqp.support.converter.Jackson2JsonMessag
   eConverter;
import
   org.springframework.amqp.support.converter.MessageConverter;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class MQConfig {

             public    static    final    String    QUEUE1    =
   "2ite_micro_message_queue1";
             public    static    final    String    QUEUE2    =
   "2ite_micro_message_queue2";
           public    static    final    String    EXCHANGE    =
   "2ite_micro_message_exchange2";
         public    static    final    String    ROUTING_KEY1    =
   "message_routingKey1";
          public    static    final    String    ROUTING_KEY2    =
   "message_routingKey2";

    @Bean
    public Queue queue1() {
        return  new Queue(QUEUE1);
    }

    @Bean
    public Queue queue2() {
        return  new Queue(QUEUE2);
    }

    @Bean
    public TopicExchange exchange() {
        return new TopicExchange(EXCHANGE);
    }

    @Bean
         public  Binding  binding1(Queue  queue1,  TopicExchange
   exchange) {
        return BindingBuilder
                .bind(queue1)
                .to(exchange)
                .with(ROUTING_KEY1);
```

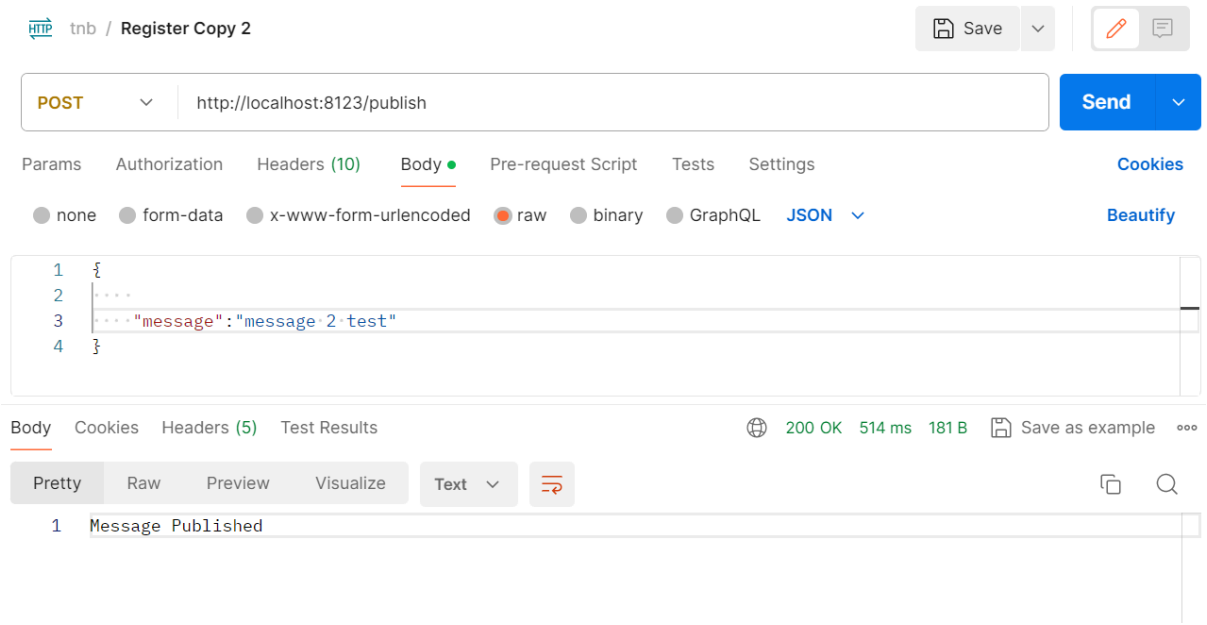
```
    @Bean
        public Binding binding2(Queue queue2, TopicExchange
exchange) {
        return BindingBuilder
                .bind(queue2)
                .to(exchange)
                .with(ROUTING_KEY2);
    }

    @Bean
    public MessageConverter messageConverter() {
        return  new Jackson2JsonMessageConverter();
    }

    @Bean
            public AmqpTemplate template(ConnectionFactory
connectionFactory) {
                        RabbitTemplate template = new
RabbitTemplate(connectionFactory);
        template.setMessageConverter(messageConverter());
        return  template;
    }

}
```

- CustomMessage avec les 2 routing keys

```
package com.oussama.rabbitmicro;

import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import java.util.Date;
import java.util.UUID;

@RestController
public class MessagePublisher {

    @Autowired
    private RabbitTemplate template;
```

```
    @PostMapping("/publish")
    public String publishMessage(@RequestBody CustomMessage message) {
        message.setMessageId(UUID.randomUUID().toString());
        message.setMessageDate(new Date());
        template.convertAndSend(MQConfig.EXCHANGE,
                MQConfig.ROUTING_KEY1, message);
        template.convertAndSend(MQConfig.EXCHANGE,
                MQConfig.ROUTING_KEY2, message);


        return "Message Published";
    }
}
```

➔ Test avec Postman:



RabbitMQ Verification

Virtual host	Name	Type	Features	Message rate in	Message rate out	+/-
/	(AMQP default)	direct	D			
/	2iteExchange	direct	D			
/	2ite_micro_message_exchange	topic	D			
/	2ite_micro_message_exchange2	topic	D	0.00/s	0.00/s	
/	amq.direct	direct	D			
/	amq.fanout	fanout	D			
/	amq.headers	headers	D			
/	amq.match	headers	D			
/	amq.rabbitmq.trace	topic	D I			
/	amq.topic	topic	D			
/	user.exchange	direct	D			

Overview					Messages			Message rates			+/-
Virtual host	Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
/	2iteQueue	classic	D Args	idle	1	0	1				
/	2ite_micro_message_queue	classic	D	idle	0	0	0				
/	2ite_micro_message_queue1	classic	D	idle	1	0	1	0.00/s	0.00/s	0.00/s	
/	2ite_micro_message_queue2	classic	D	idle	1	0	1	0.00/s			
/	user.queue	classic	D	idle	0	0	0				

Queue 2ite_micro_message_queue2

Message 1

The server reported 0 messages remaining.

Exchange: 2ite_micro_message_exchange2

Routing Key: message_routingKey2

Redelivered: ○

Properties:

priority: 0
delivery_mode: 2
headers: __TypeId__: com.oussama.rabbitmicro.CustomMessage
content_encoding: UTF-8
content_type: application/json

Payload
107 bytes
Encoding: string

{"messageId":"12e6b58c-0725-405f-aa85-aad01b819e59","message":"message 2 test","messageDate":1705652819142}

Queue2ite_micro_message_queue1

Get Message(s)

Message 1

The server reported 0 messages remaining.

Exchange	2ite_micro_message_exchange2
Routing Key	message_routingKey1
Redelivered	•
Properties	priority: 0 delivery_mode: 2 headers: __TypeId__: com.oussama.rabbitmicro.CustomMessage content_encoding: UTF-8 content_type: application/json
Payload 107 bytes Encoding: string	{"messageId":"12e6b58c-0725-405f-aa85-aad01b819e59","message":"message 2 test","messageDate":1705652819142}

Partie 3: spring-rabbitmq-cosomer consommateurs 1 et 2

- Microservices_RabbitMq_Messagerie-main
  - spring-rabbitmq-consumer [boot]
    - src/main/java
      - com.oussama.rabbitmicro
        - CustomMessage.java
        - MessageListener.java
        - MQConfig.java
        - SpringRabbitmqConsumerApplication.java
    - src/main/resources
    - src/test/java
    - JRE System Library [JavaSE-11]
    - Maven Dependencies
    - target/generated-sources/annotations
    - target/generated-test-sources/test-annotations
    - src
    - target
    - mvnw
    - mvnw.cmd
    - pom.xml

Configuration du consommateur 1:

```
package com.oussama.rabbitmicro;


import org.springframework.amqp.core.*;
import
   org.springframework.amqp.rabbit.connection.ConnectionFactory;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import
   org.springframework.amqp.support.converter.Jackson2JsonMessag
   eConverter;
import
   org.springframework.amqp.support.converter.MessageConverter;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
```

```
@Configuration
public class MQConfig {

            public static final String QUEUE1 =
"2ite_micro_message_queue1";
          public static final String EXCHANGE =
"2ite_micro_message_exchange";
         public static final String ROUTING_KEY1 =
"message_routingKey1";

 @Bean
 public Queue queue1() {
     return  new Queue(QUEUE1);
 }


 @Bean
 public TopicExchange exchange() {
     return new TopicExchange(EXCHANGE);
 }

 @Bean
     public Binding binding1(Queue queue1, TopicExchange
exchange) {
     return BindingBuilder
             .bind(queue1)
             .to(exchange)
             .with(ROUTING_KEY1);
 }



 @Bean
 public MessageConverter messageConverter() {
     return  new Jackson2JsonMessageConverter();
 }

 @Bean
          public AmqpTemplate template(ConnectionFactory
connectionFactory) {
                        RabbitTemplate template = new
RabbitTemplate(connectionFactory);
     template.setMessageConverter(messageConverter());
     return  template;
 }
```

MessageListener du consommateur 1:

```
package com.oussama.rabbitmicro;
import org.springframework.amqp.rabbit.annotation.RabbitListener;
import org.springframework.stereotype.Component;
@Component
public class MessageListener {
   @RabbitListener(queues = MQConfig.QUEUE1)
   public void listener(CustomMessage message) {
       System.out.println(message);
   }
}
```

Configuration du consommateur 2:

```
package com.oussama.rabbitmicro;

import org.springframework.amqp.core.*;
import org.springframework.amqp.rabbit.connection.ConnectionFactory;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import
org.springframework.amqp.support.converter.Jackson2JsonMessageConverter;
import org.springframework.amqp.support.converter.MessageConverter;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class MQConfig {

    public static final String QUEUE2 = "2ite_micro_message_queue2";
    public static final String EXCHANGE2 = "2ite_micro_message_exchange";
    public static final String ROUTING_KEY2 = "message_routingKey2";

    @Bean
    public Queue queue1() {
        return  new Queue(QUEUE2);
    }


    @Bean
    public TopicExchange exchange() {
        return new TopicExchange(EXCHANGE2);
    }

    @Bean
    public Binding binding1(Queue queue2, TopicExchange exchange2) {
        return BindingBuilder
                .bind(queue2)
                .to(exchange2)
                .with(ROUTING_KEY2);
    }
```

```
    @Bean
    public MessageConverter messageConverter() {
        return  new Jackson2JsonMessageConverter();
    }

    @Bean
    public AmqpTemplate template(ConnectionFactory connectionFactory) {
        RabbitTemplate template = new RabbitTemplate(connectionFactory);
        template.setMessageConverter(messageConverter());
        return  template;
    }

}
```

MessageListener du consommateur 2:

```
package com.oussama.rabbitmicro;

import org.springframework.amqp.rabbit.annotation.RabbitListener;
import org.springframework.stereotype.Component;

@Component
public class MessageListener {

    @RabbitListener(queues = MQConfig.QUEUE2)
    public void listener(CustomMessage message) {
        System.out.println(message);
    }

}
```