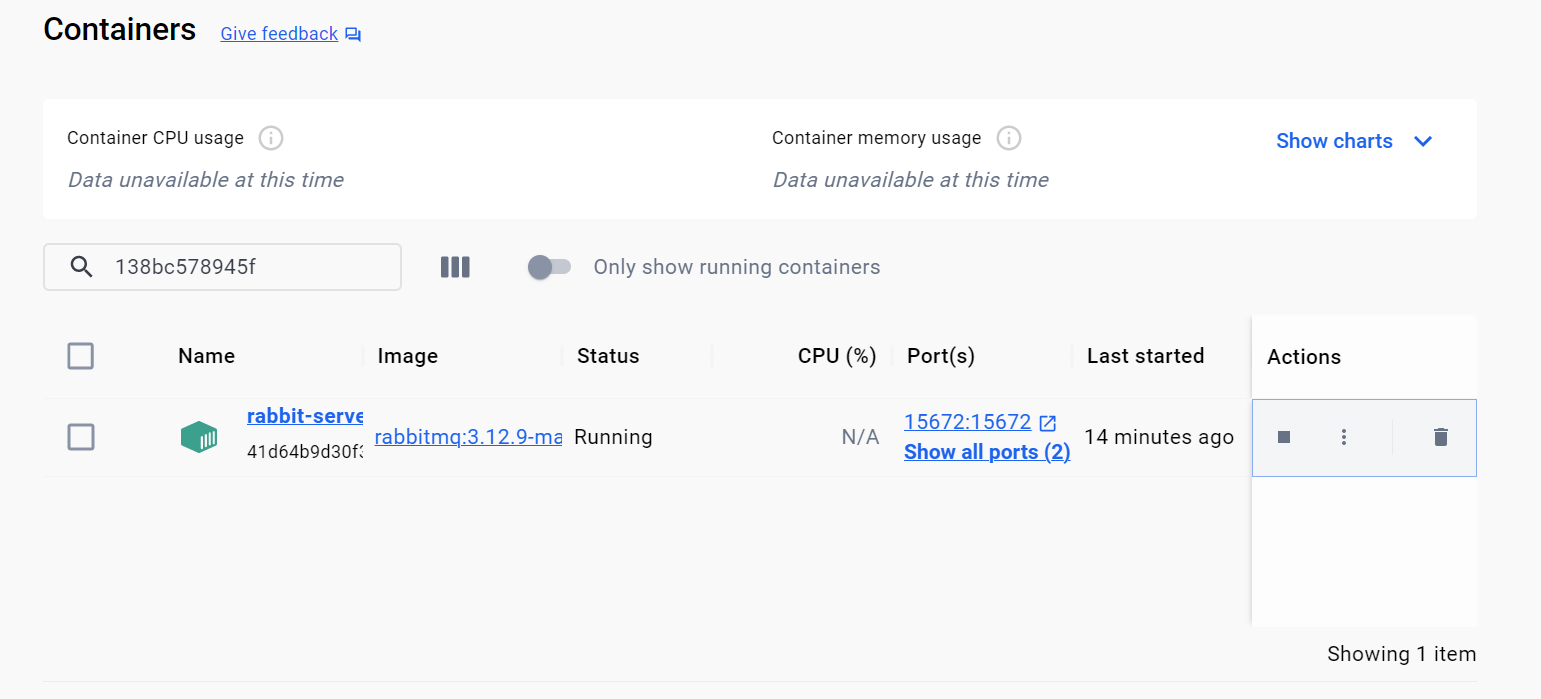
TP: RabbitMQ (Un producteur et deux consummer )

TP réalisé par afsa Karchaou (5iir G7)TP:

Partie 1:

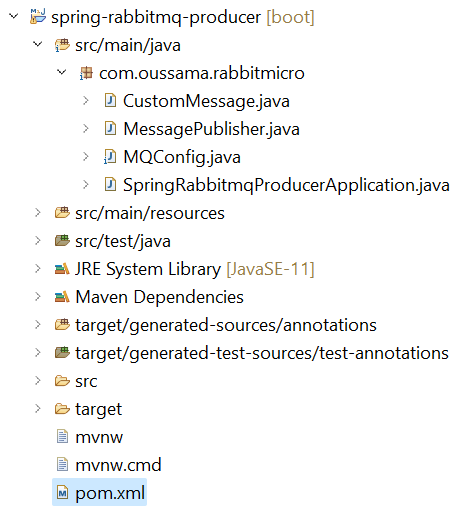
Installation de l’image Docker:

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.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 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,

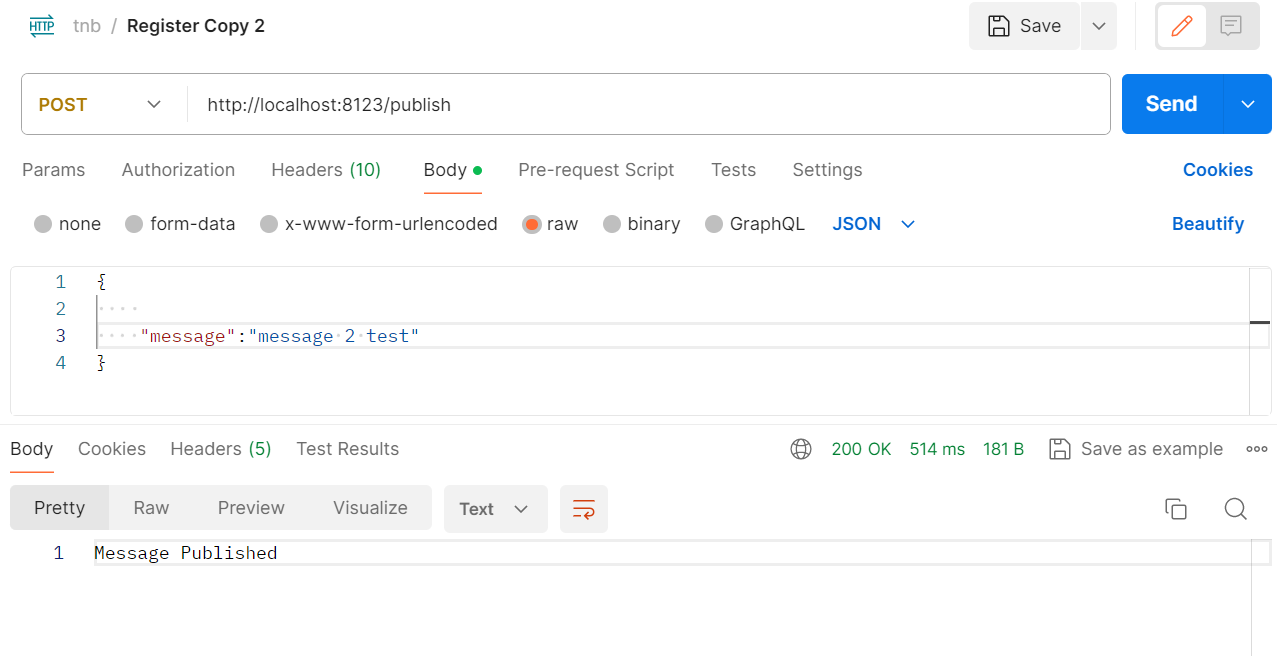
MQConfig.ROUTING\_KEY2, message);

return "Message Published";

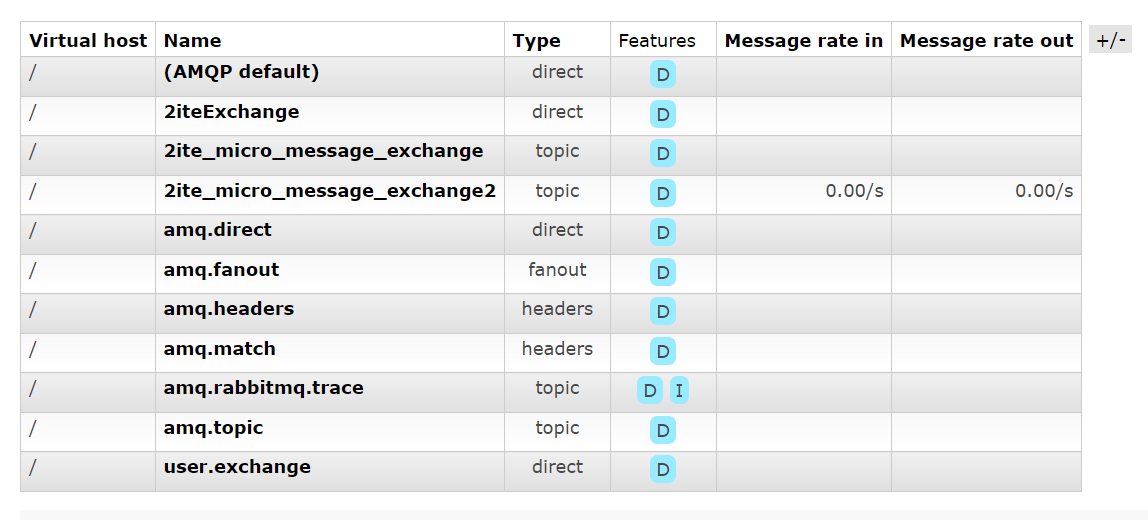
}

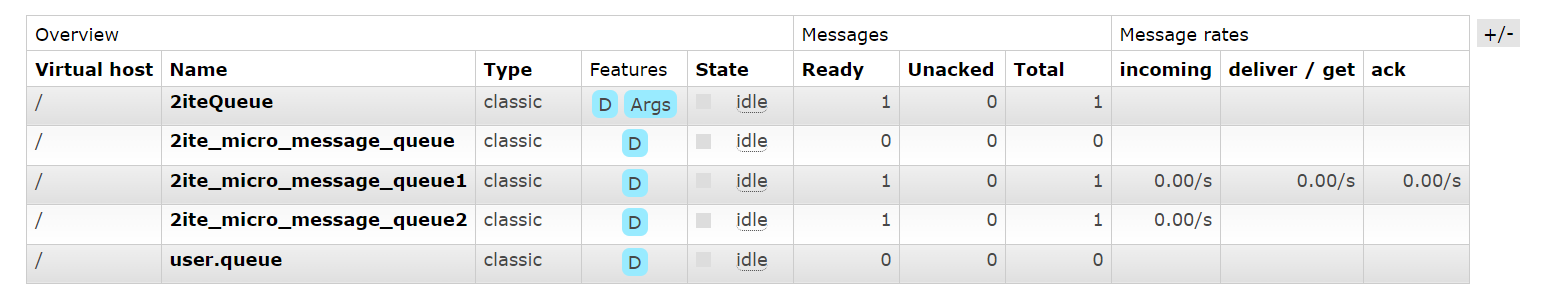
}

* Test avec Postman:

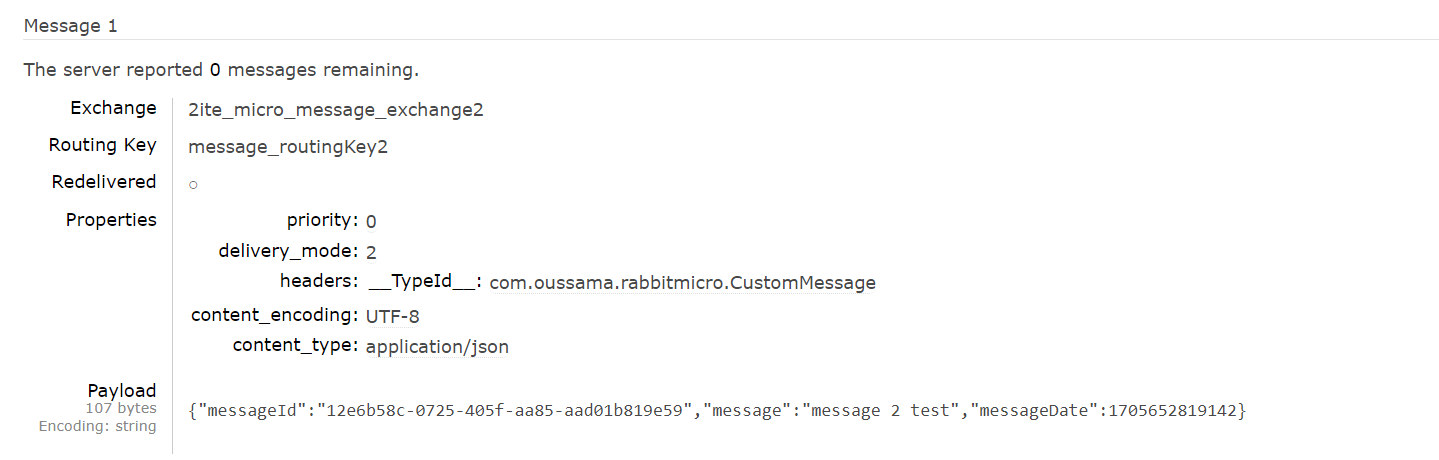


* Vérification





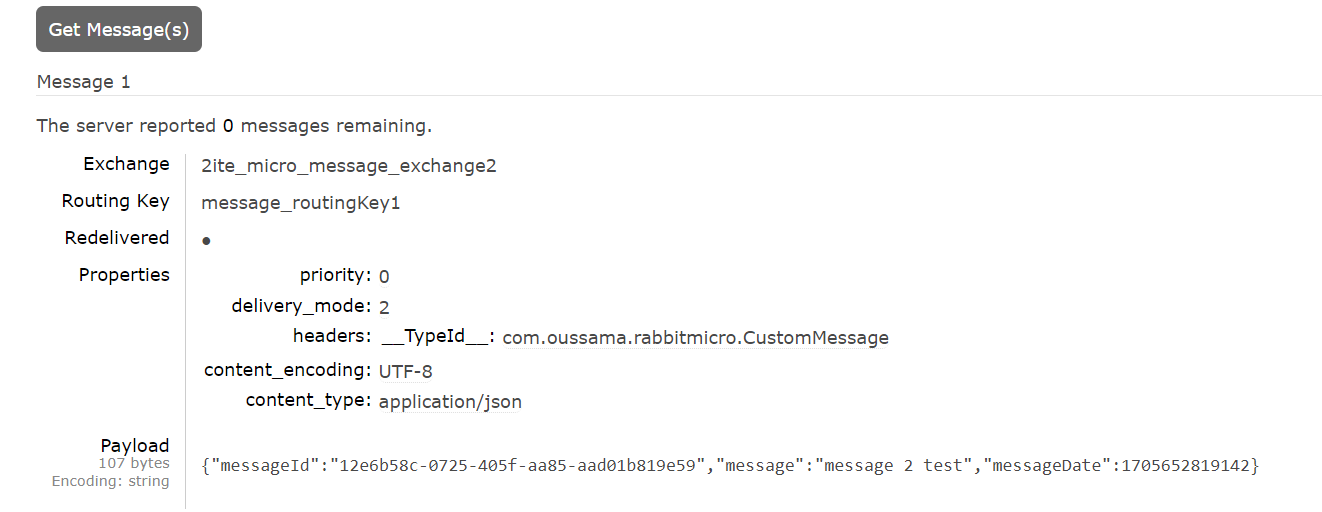
# Queue 2ite\_micro\_message\_queue2



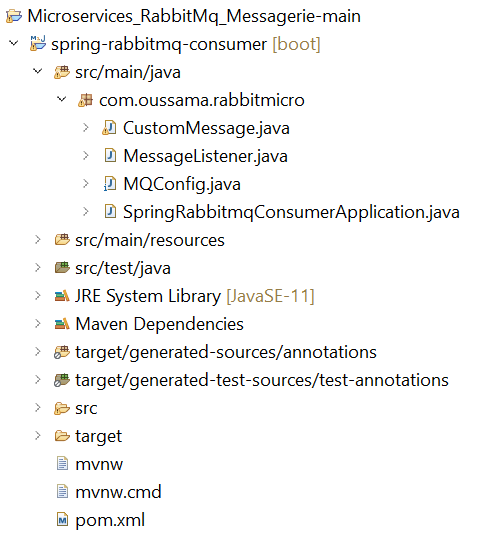
# 

# 

# Queue2ite\_micro\_message\_queue1



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



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.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 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);

}

}