Here is the `RabbitMQConfig` class for both microservices based on your use case for "panier" and "commande."

For Panier Microservice (`RabbitMQConfig.java`):

```
package com.example.Panier.configuration;

import org.springframework.amqp.core.Binding;
import org.springframework.amqp.core.BindingBuilder;
import org.springframework.amqp.core.Exchange;
import org.springframework.amqp.core.ExchangeBuilder;
import org.springframework.amqp.core.Queue;
import org.springframework.amqp.core.QueueBuilder;
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 RabbitMQConfig {

    // Declare the direct exchange
    @Bean
    public Exchange sagaExchange() {
        return ExchangeBuilder.directExchange("saga-exchange").build();
    }

    // Message Converter for JSON
    @Bean
    public MessageConverter converter() {
        return new Jackson2JsonMessageConverter();
    }

    // QUEUES
    @Bean
    public Queue panierApiGetProducerQueue() {
        return QueueBuilder.durable("panier-api-get-producer-queue").build();
    }

    @Bean
    public Queue panierApiGetConsumerQueue() {
        return QueueBuilder.durable("panier-api-get-consumer-queue").build();
    }

    @Bean
    public Queue panierApiPostProducerQueue() {
        return QueueBuilder.durable("panier-api-post-producer-queue").build();
    }

    @Bean
    public Queue panierApiPostConsumerQueue() {
        return QueueBuilder.durable("panier-api-post-consumer-queue").build();
    }

    // BINDING
    @Bean
    public Binding panierApiGetProducerBinding() {
        return
BindingBuilder.bind(panierApiGetProducerQueue()).to(sagaExchange())
                .with("panier-api-get-producer-routing-key").noargs();
    }

    @Bean
    public Binding panierApiGetConsumerBinding() {
        return
BindingBuilder.bind(panierApiGetConsumerQueue()).to(sagaExchange())
                .with("panier-api-get-consumer-routing-key").noargs();
    }

    @Bean
    public Binding panierApiPostProducerBinding() {
        return
BindingBuilder.bind(panierApiPostProducerQueue()).to(sagaExchange())
                .with("panier-api-post-producer-routing-key").noargs();
    }

    @Bean
    public Binding panierApiPostConsumerBinding() {
        return
BindingBuilder.bind(panierApiPostConsumerQueue()).to(sagaExchange())
                .with("panier-api-post-consumer-routing-key").noargs();
    }
}
```

For Commande Microservice (`RabbitMQConfig.java`):

```
package com.example.Commande.configuration;

import org.springframework.amqp.core.Binding;
import org.springframework.amqp.core.BindingBuilder;
import org.springframework.amqp.core.Exchange;
import org.springframework.amqp.core.ExchangeBuilder;
import org.springframework.amqp.core.Queue;
import org.springframework.amqp.core.QueueBuilder;
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 RabbitMQConfig {

    // Declare the direct exchange
    @Bean
    public Exchange sagaExchange() {
        return ExchangeBuilder.directExchange("saga-exchange").build();
    }

    // Message Converter for JSON
    @Bean
    public MessageConverter converter() {
        return new Jackson2JsonMessageConverter();
    }

    // QUEUES
    @Bean
    public Queue commandeApiGetProducerQueue() {
        return QueueBuilder.durable("commande-api-get-producer-
queue").build();
    }

    @Bean
    public Queue commandeApiGetConsumerQueue() {
        return QueueBuilder.durable("commande-api-get-consumer-
queue").build();
    }

    @Bean
    public Queue commandeApiPostProducerQueue() {
        return QueueBuilder.durable("commande-api-post-producer-
queue").build();
    }

    @Bean
    public Queue commandeApiPostConsumerQueue() {
        return QueueBuilder.durable("commande-api-post-consumer-
queue").build();
    }

    @Bean
    public Queue compensatePanierQueue() {
        return QueueBuilder.durable("compensate-panier-queue").build();
    }

    // BINDING
    @Bean
    public Binding commandeApiGetProducerBinding() {
        return
BindingBuilder.bind(commandeApiGetProducerQueue()).to(sagaExchange())
                .with("commande-api-get-producer-routing-key").noargs();
    }

    @Bean
    public Binding commandeApiGetConsumerBinding() {
        return
BindingBuilder.bind(commandeApiGetConsumerQueue()).to(sagaExchange())
                .with("commande-api-get-consumer-routing-key").noargs();
    }

    @Bean
    public Binding commandeApiPostProducerBinding() {
        return
BindingBuilder.bind(commandeApiPostProducerQueue()).to(sagaExchange())
                .with("commande-api-post-producer-routing-key").noargs();
    }

    @Bean
    public Binding commandeApiPostConsumerBinding() {
        return
BindingBuilder.bind(commandeApiPostConsumerQueue()).to(sagaExchange())
                .with("commande-api-post-consumer-routing-key").noargs();
    }

    @Bean
    public Binding compensatePanierBinding() {
        return BindingBuilder.bind(compensatePanierQueue()).to(sagaExchange())
                .with("compensate-panier-routing-key").noargs();
    }
}
```

Breakdown:

1. **Queues**:
   - `panier-api-get-producer-queue` and `panier-api-get-consumer-queue` for getting panier details (GET request).
   - `panier-api-post-producer-queue` and `panier-api-post-consumer-queue` for posting panier updates (POST request).
   - `commande-api-get-producer-queue` and `commande-api-get-consumer-queue` for getting commande details (GET request).
   - `commande-api-post-producer-queue` and `commande-api-post-consumer-queue` for posting commande updates (POST request).
   - `compensate-panier-queue` for compensating the panier in case of failure.
2. **Bindings**:
   - Each queue is bound to the exchange with a specific routing key to match messages between services in the saga.

With this setup, the services will be able to communicate through RabbitMQ as per the saga pattern, ensuring proper message flow between services.