Spring Integration: Using Channel Adapters to Integrate with External Systems

INTEGRATING WITH APACHE KAFKA

Steven Haines
PRINCIPAL SOFTWARE ARCHITECT
@geekcap www.geekcap.com

Overview

Introduction to Apache Kafka

Inbound and Outbound Channel Adapters

Inbound and Outbound Gateways

Apache Kafka

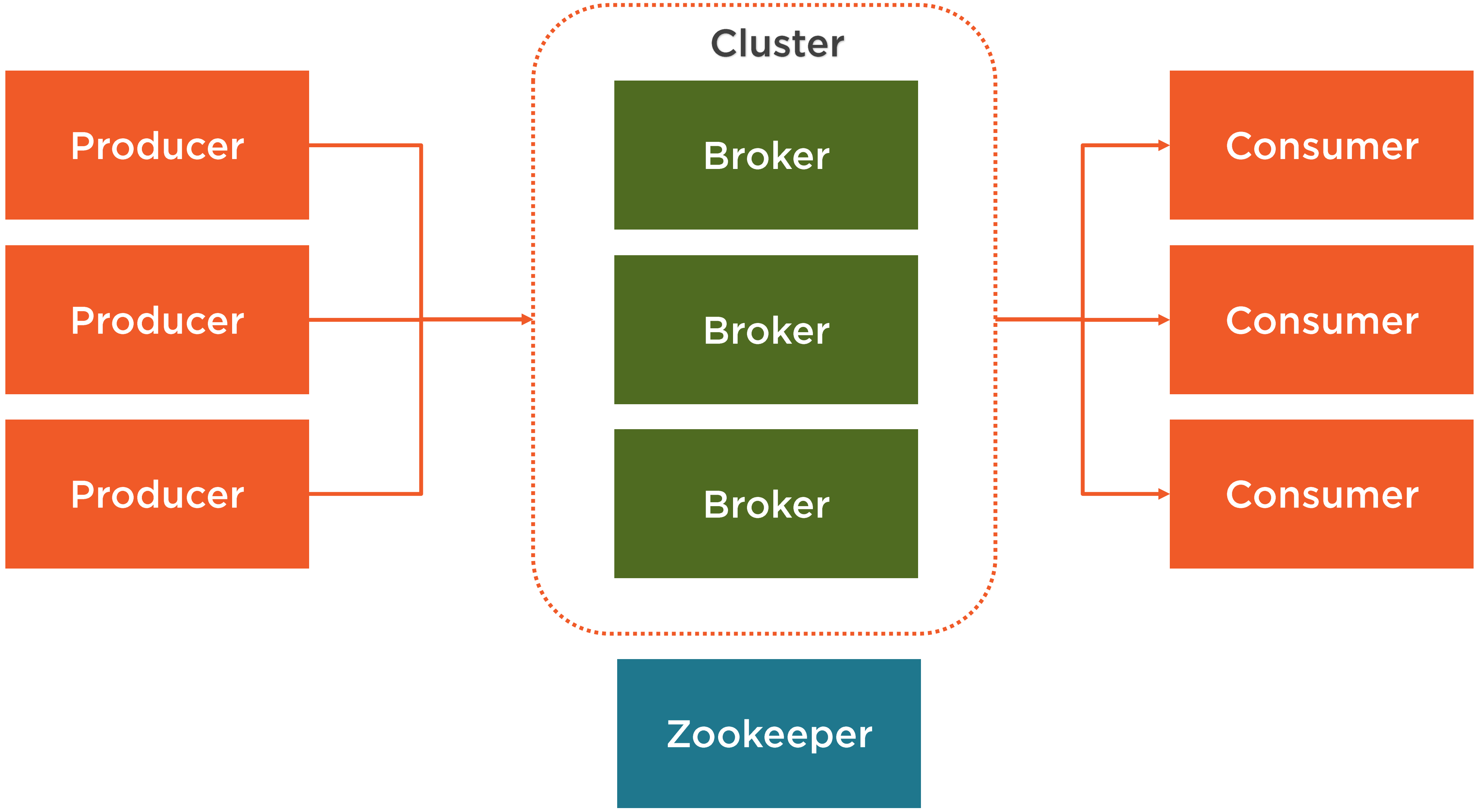
Kafka is a publish-subscribe based durable distributed streaming platform.

Use Case

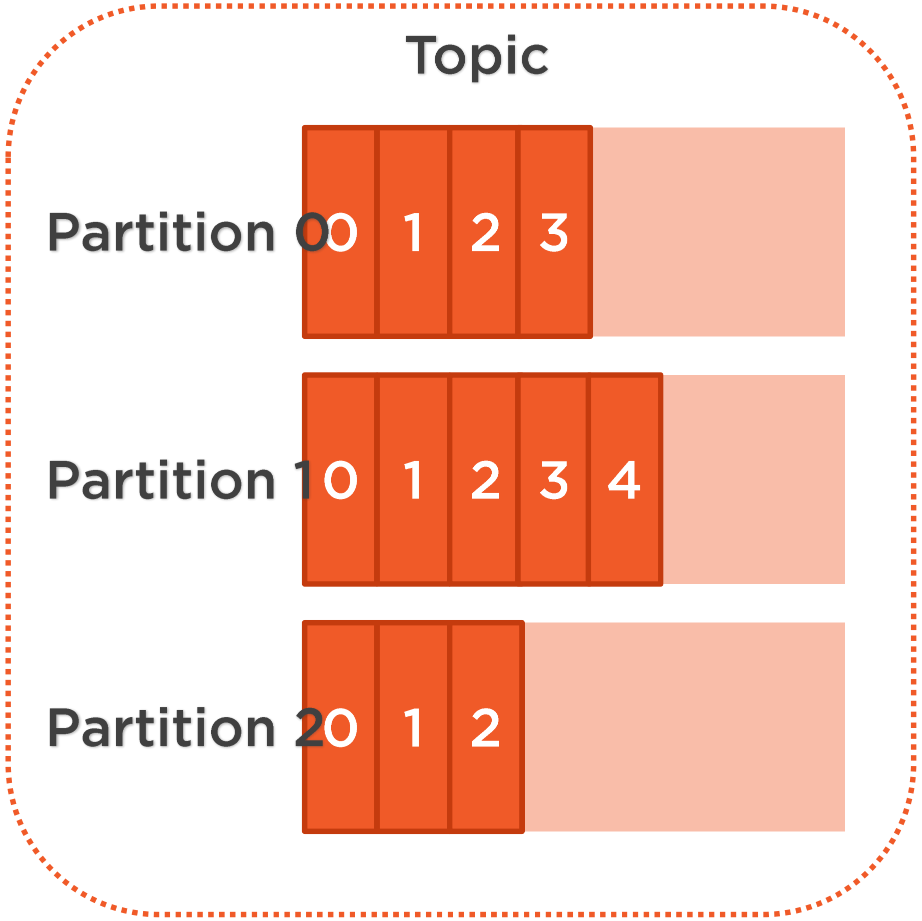
Building real-time streaming data pipelines that reliably get data between systems or applications

Building real-time streaming applications that transform or react to the streams of data

Apache Kafka High-level Architecture



Apache Kafka Topics



Apache Kafka – Inbound and Outbound Channel Adapters

Inbound and Outbound Channel Adapters

Inbound Channel Adapter

KafkaMessageSource

Outbound Channel Adapter

KafkaProducerMessageHandle
r

```
@Configuration
public class KafkaInboundConfig {

    @Bean
    public MessageChannel
reservationFromKafka() {
        return new DirectChannel();
    }

    @InboundChannelAdapter(
                    channel =
"reservationFromKafka",
                    poller =
@Poller(fixedDelay = "1000"))
    @Bean
    public KafkaMessageSource<String, String>

kafkaSource(ConsumerFactory<String, String>
cf) {
        ConsumerProperties consumerProperties
=
                        new
ConsumerProperties("reservationTopic");

consumerProperties.setGroupId("reservationGrou
p");
```

- Setup a configuration class and enable Spring Integration
- Define a MessageChannel
- Create an InboundChannelAdapter
- Set the topic name and group name in a ConsumerProperties instance
- Create the KafkaMessageSource with the ConsumerFactory and ConsumerProperties

```
@Bean
public ProducerFactory<String, String>
producerFactory() {
    Map<String, Object> props = new HashMap<>();
    props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
brokerAddress);
    props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
                    StringSerializer.class.getName());
    props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
                    StringSerializer.class.getName());
    return new DefaultKafkaProducerFactory<>(props);
}

@Bean
public KafkaTemplate<String, String>
template() {
    return new
KafkaTemplate<>(producerFactory());
}

@Bean
@ServiceActivator(inputChannel = “toKafka")
 public MessageHandler handler() throws
Exception {
    KafkaProducerMessageHandler<String,
String> handler =
            new
KafkaProducerMessageHandler<>(template());
   handler.setTopicExpression(new
```

◂ Create a ProducerFactory

◂ Set the Broker Address

◂ Set the key and value serializers

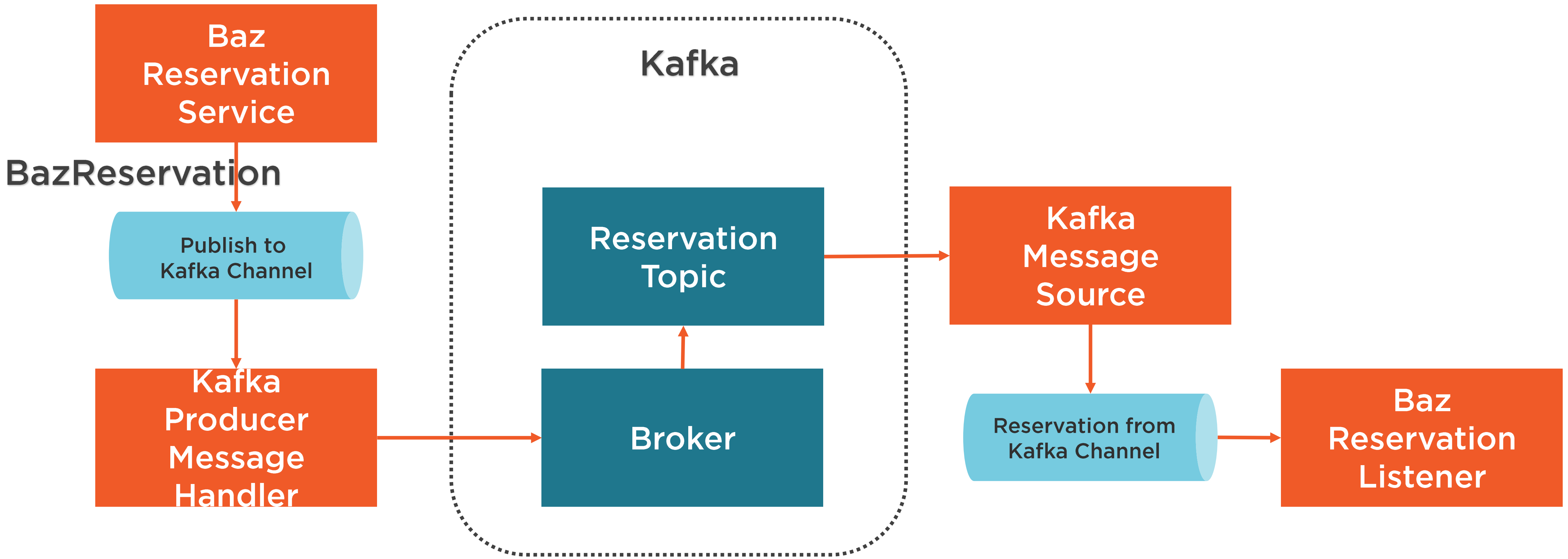
◂ Create a KafkaTemplate that uses the producerFactory

◂ Create a KafkaProducerMessageHandler

◂ Specify the topic name

◂ Specify the consumer key

Example: Reservation Service



```
git clone
https://github.com/wurstmeister/kafka-
docker.git

docker-compose -f docker-compose-single-
broker.yml up -d
```

Running Kafka in Docker

Kafka Docker Image: wurstmeister/kafka

Kafka DockerHub Link: https://hub.docker.com/r/wurstmeister/kafka

Demo

Build our applications

- Baz Reservation Publisher
- Kafka Globomantics Registration Service

Kafka – Inbound and Outbound Gateways

Inbound and Outbound Gateways

Inbound Gateway

KafkaInboundGateway

Outbound Gateway

KafkaProducerMessageHandle
r

```
@Bean
public KafkaMessageListenerContainer
container(

ConsumerFactory consumerFactory) {
    ContainerProperties containerProperties =
                        new
ContainerProperties("addressTopic");

containerProperties.setGroupId("addressGroup")
;
    return new KafkaMessageListenerContainer(

consumerFactory, containerProperties);
}

@Bean
public KafkaInboundGateway<String, String,
String> in(

AbstractMessageListenerContainer<String,
String> c,
              KafkaTemplate<String, String>
replyTemplate) {

replyTemplate.setDefaultTopic("addressReplyTop
```

- Create a KafkaMessageListenerContainer that listens for messages on the addressTopic
- Create a KafkaInboundGateway
- Set reply topic
- Set request and reply channels

```
@Bean
public KafkaMessageListenerContainer container(ConsumerFactory
consumerFactory) {
    ContainerProperties containerProperties = new
ContainerProperties("addressReplyTopic");
    containerProperties.setGroupId("addressGroup");
    return new KafkaMessageListenerContainer(consumerFactory,
containerProperties);
}
```

Outbound Gateway - KafkaMessageListenerContainer

Used to connect to Kafka and listen for a reply from the inbound gateway

```
@Bean
public ReplyingKafkaTemplate<String, String, String>
replyingKafkaTemplate(
            ProducerFactory<String, String> producerFactory,
            KafkaMessageListenerContainer container) {
    return new ReplyingKafkaTemplate<String, String,
String>(producerFactory, container);
 }
```

Outbound Gateway - ReplyingKafkaTemplate

Used to send a message to Kafka using the producerFactory

And to receive a reply, using the KafkaMessageListenerContainer we created in the previous slide

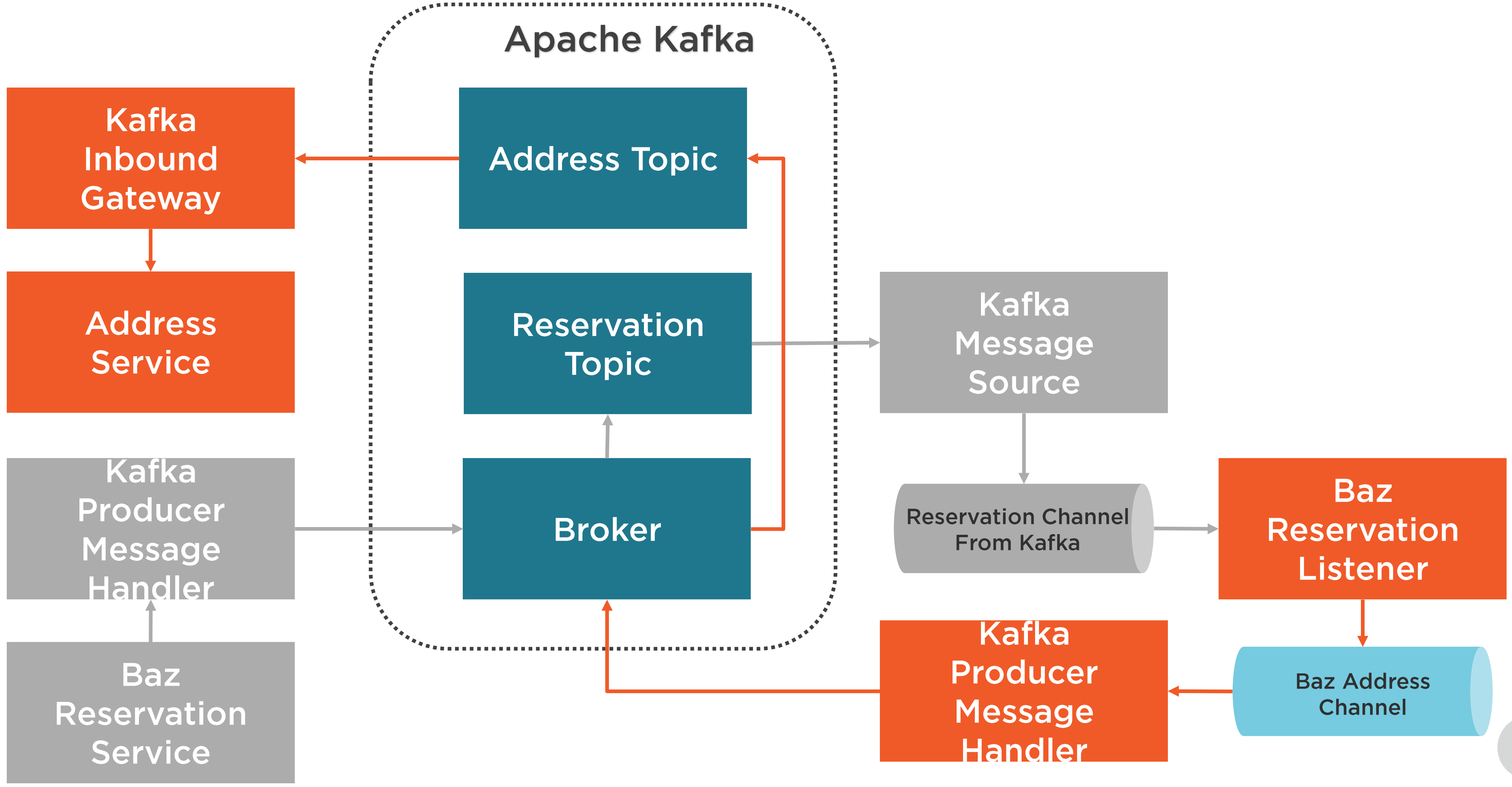
```
@Bean
@ServiceActivator(inputChannel = "bazAddressChannel")
public KafkaProducerMessageHandler<String, String> outGateway(

ReplyingKafkaTemplate<String, String, String> kafkaTemplate) {
    KafkaProducerMessageHandler<String, String> handler =
                                                            new
KafkaProducerMessageHandler<>(kafkaTemplate);
    handler.setTopicExpression(new LiteralExpression("addressTopic"));
    handler.setMessageKeyExpression(new LiteralExpression("addressKey"));
    return handler;
}
```

Outbound Gateway - KafkaProducerMessageHandler

Uses the ReplyingKafkaTemplate to send a message to the addressKey partition in the addressTopic and wait for a reply

Example: Reservation Service



Conclusion

Apache Kafka

Kafka is a publish-subscribe based durable distributed streaming platform.

Inbound and Outbound Channel Adapters

Inbound Channel Adapter

KafkaMessageSource

Outbound Channel Adapter

KafkaProducerMessageHandle
r

Inbound and Outbound Gateways

Inbound Gateway

KafkaInboundGateway

Outbound Gateway

KafkaProducerMessageHandle
r

Summary

You should understand what Apache Kafka is and what it does

You should understand how to integrate with Apache Kafka using inbound and outbound channel adapters and gateways

You should feel comfortable integrating Kafka into your own Spring Integration applications

Next Module: Integrating with Databases