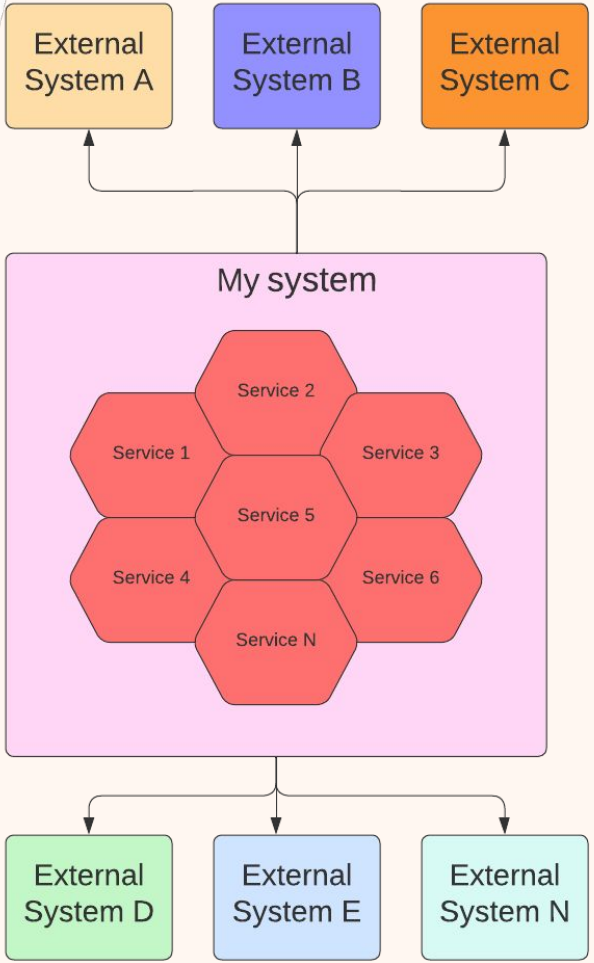
Connecting disparate systems in a lightweight way

Zineb Bendhiba

Accento October 2023

Integration challenges

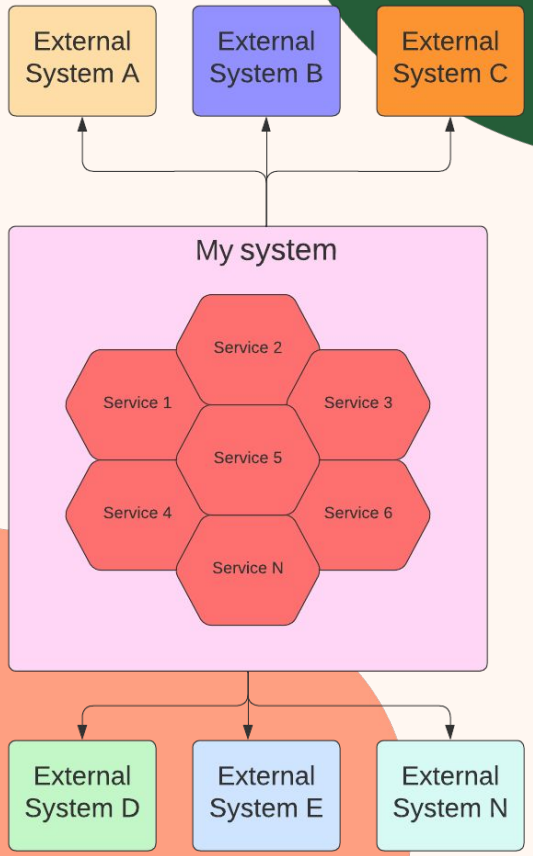


Zineb Bendhiba

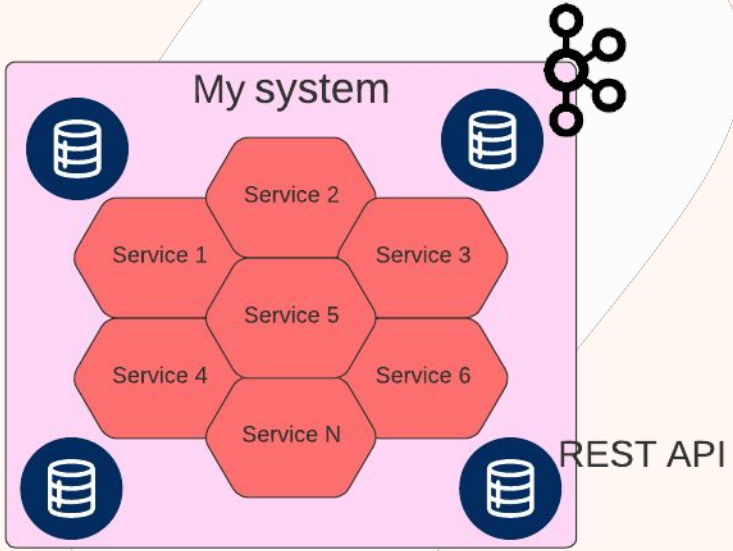
- Senior Software Engineer at Red Hat
- Apache Camel PMC
- International Speaker
- 15+ years professional software development experience
- Speak English, French, Moroccan Darija, Arabic
- https://zinebbendhiba.com

1

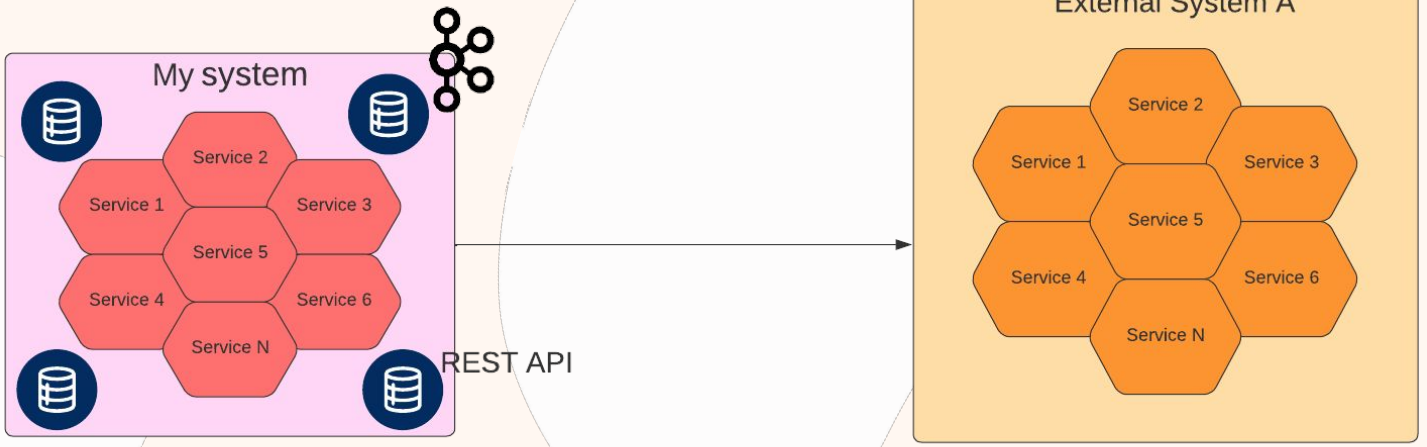
Connecting to disparate systems



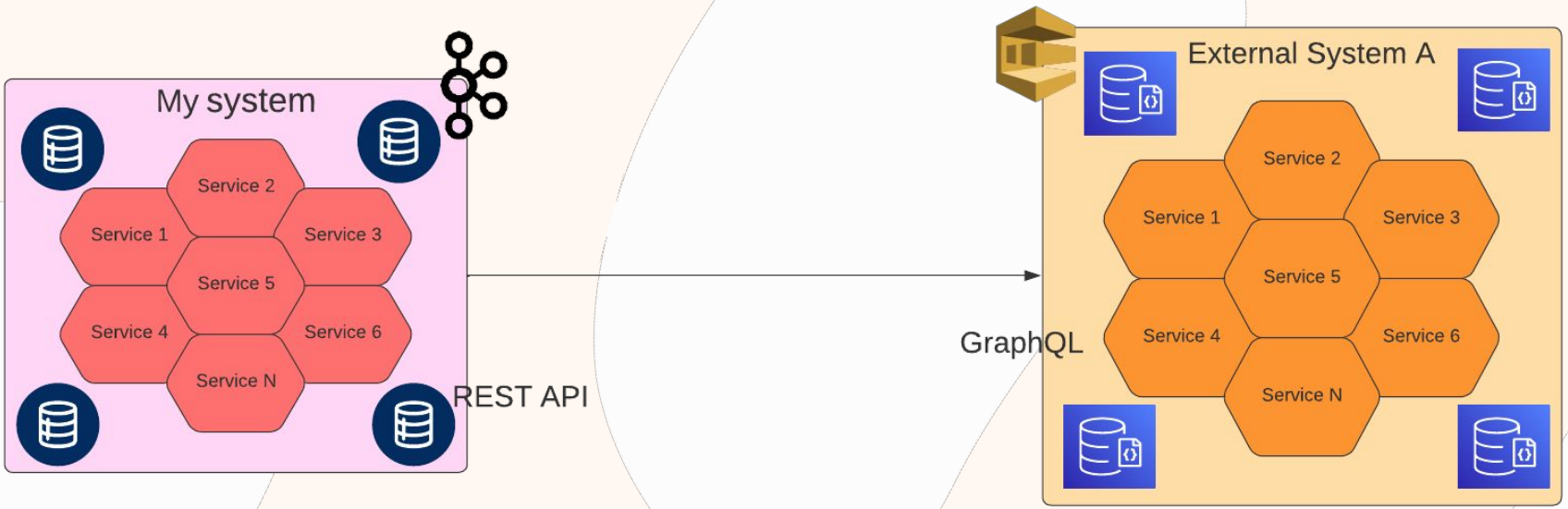
Challenge #1 : Connecting to disparate systems



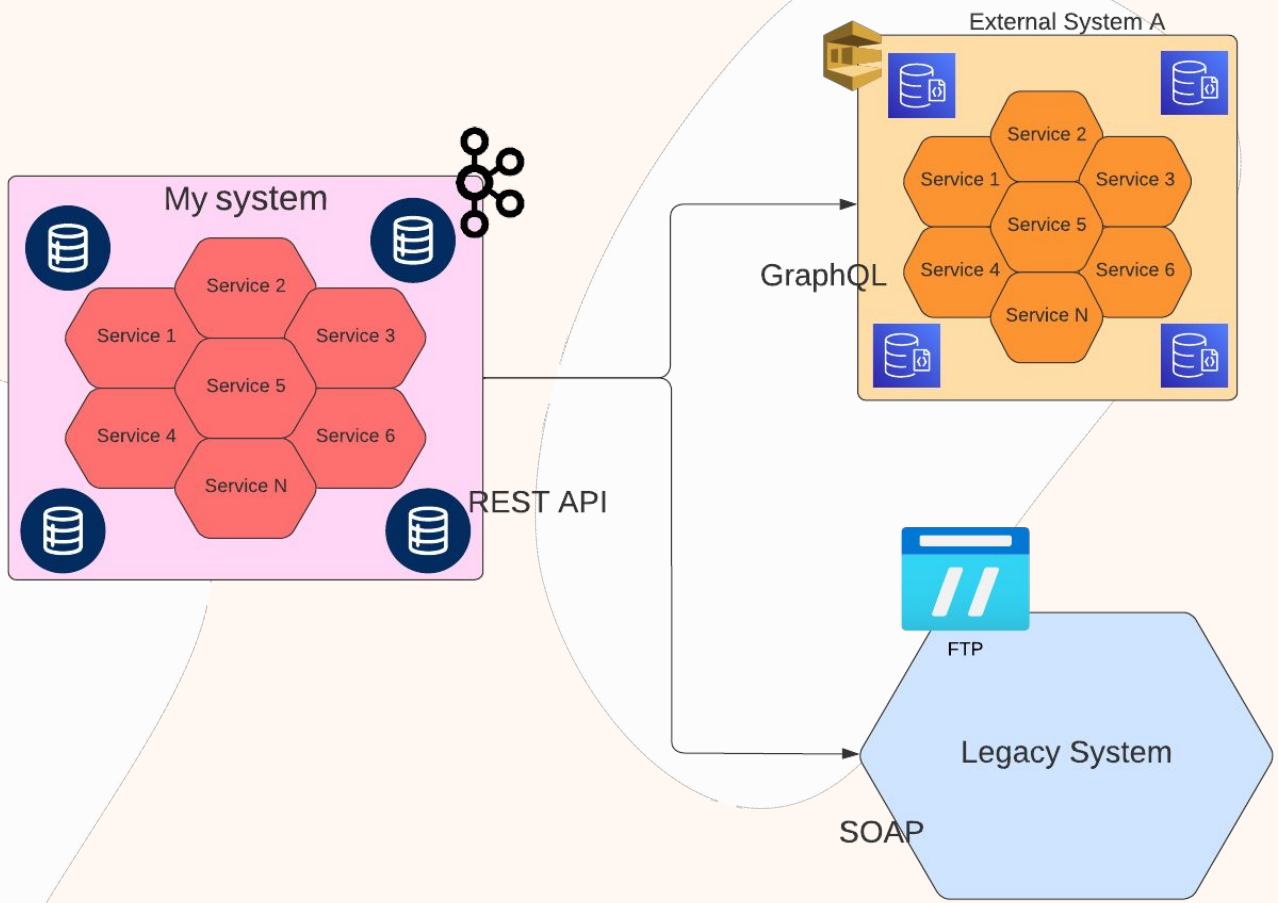
Challenge #1 : Connecting to disparate systems



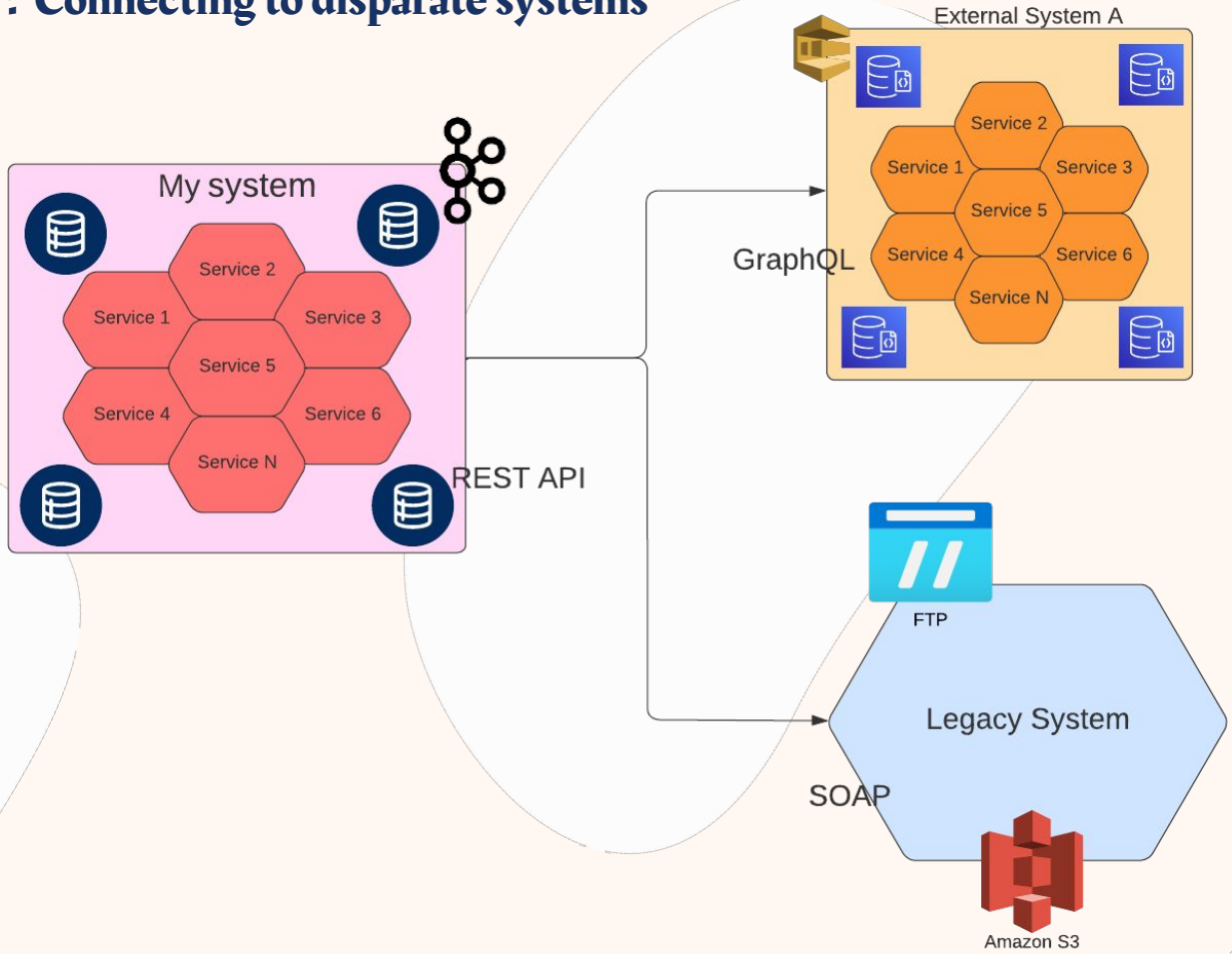
Challenge #1 : Connecting to disparate systems



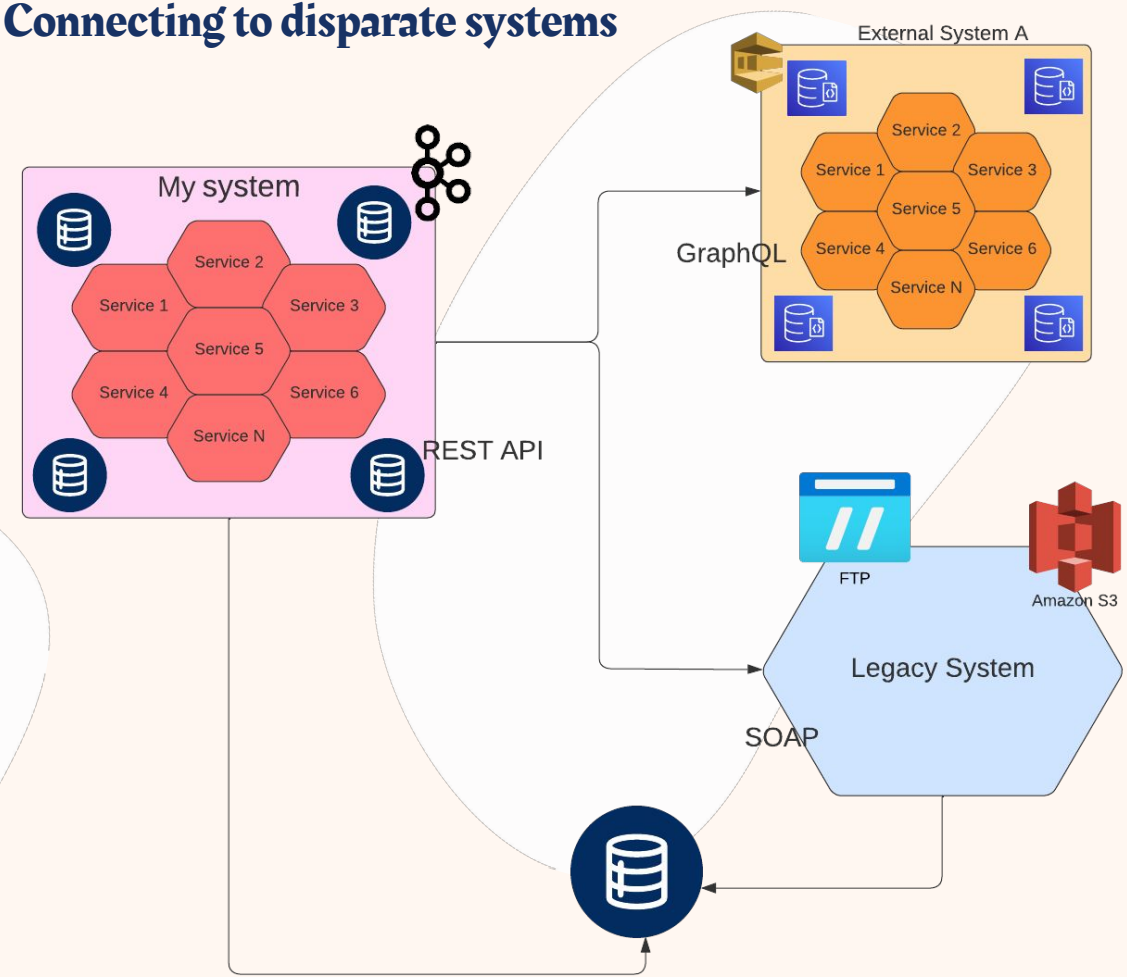
Challenge #1 : Connecting to disparate systems



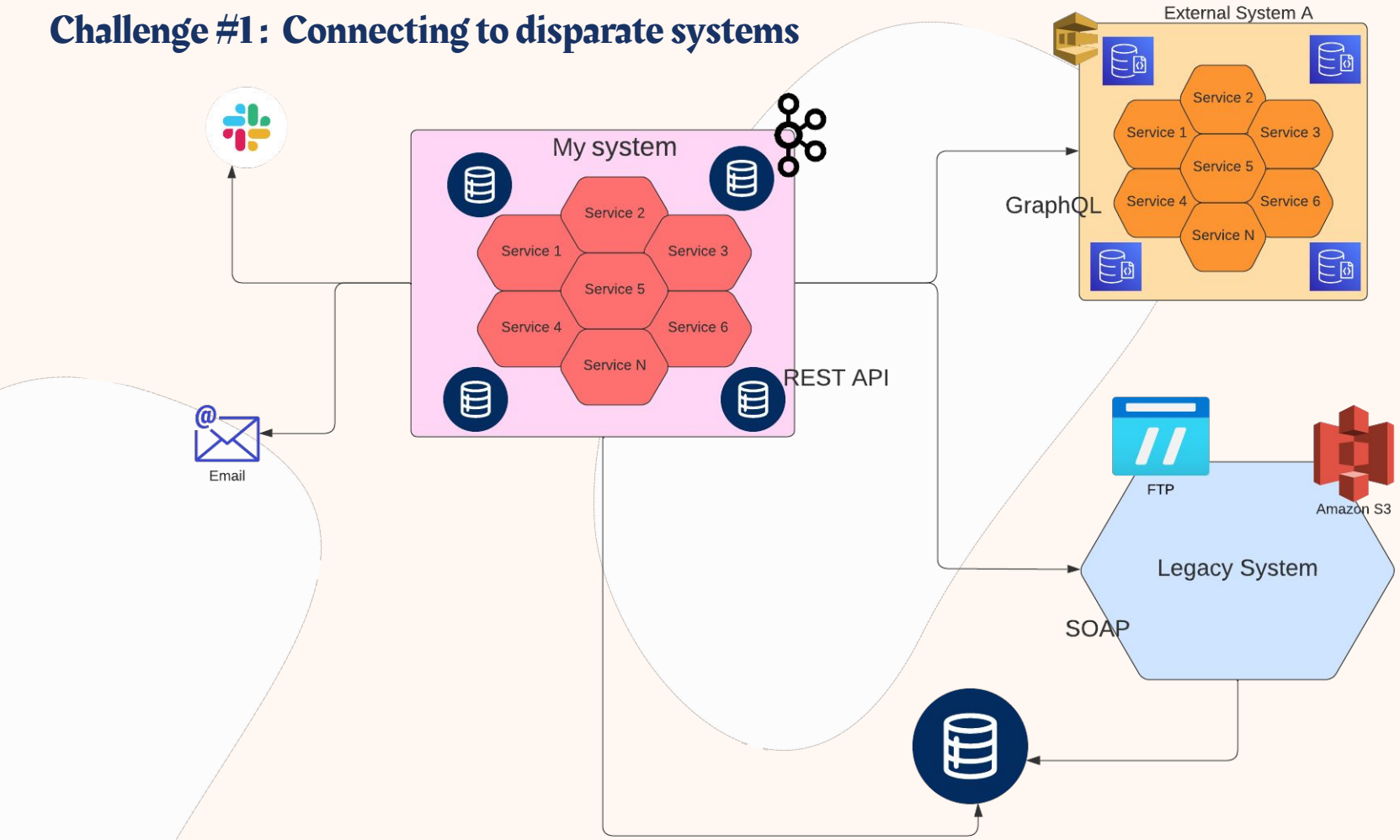
Challenge #1 : Connecting to disparate systems



Challenge #1 : Connecting to disparate systems

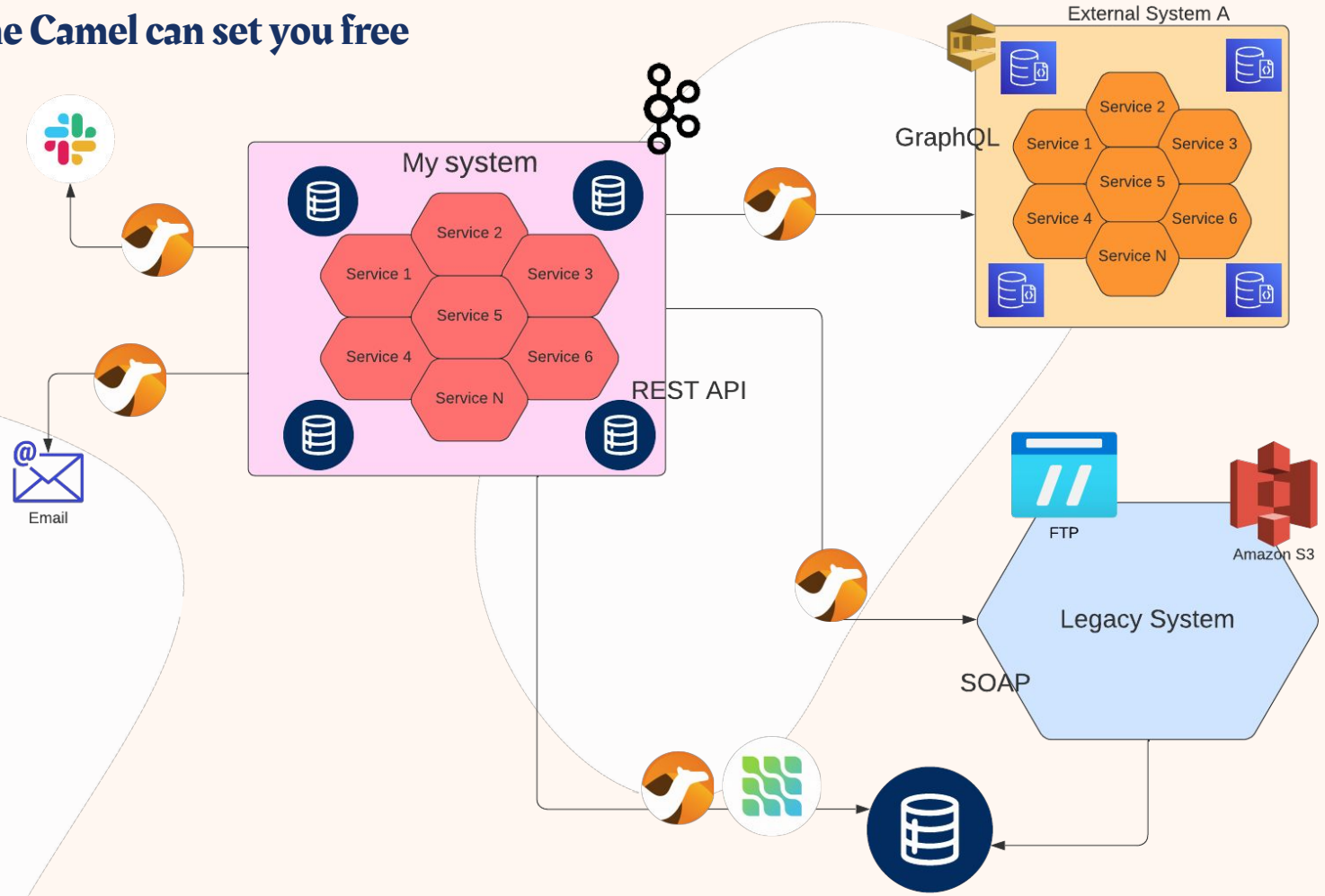


Challenge #1 : Connecting to disparate systems



Apache Camel can set you free

Apache Camel can set you free



What is Apache Camel ?

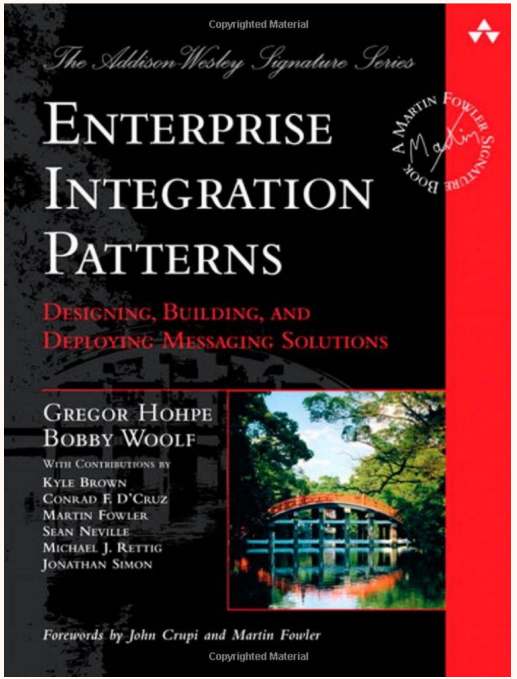
Apache Camel is an **Open Source**

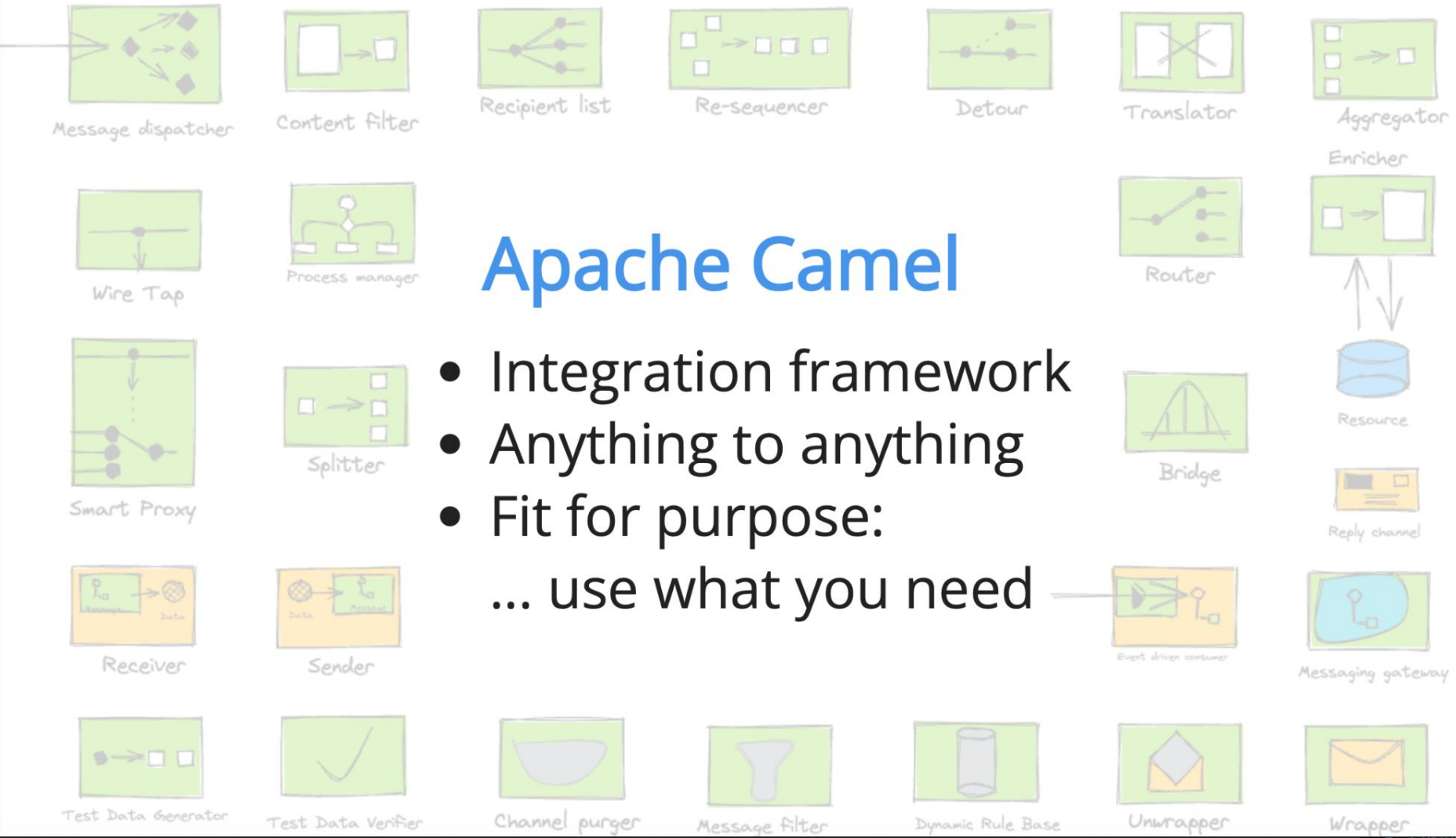
Integration Framework

Entreprise patterns

Emergent

“Design patterns” best practices





Apache Camel

- Integration framework
- Anything to anything
- Fit for purpose:
  … use what you need

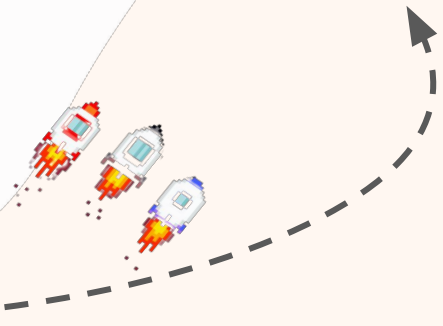
Apache Camel: Camel message Routing

Term	Meaning
Message	data transferred by a Route
Exchange	envelope; wraps the data
Endpoint	a channel, receiver or sender
Component	know-how; creates endpoints
Processor	Java API; custom logic

Apache Camel: Domain Specific Language (DSL)

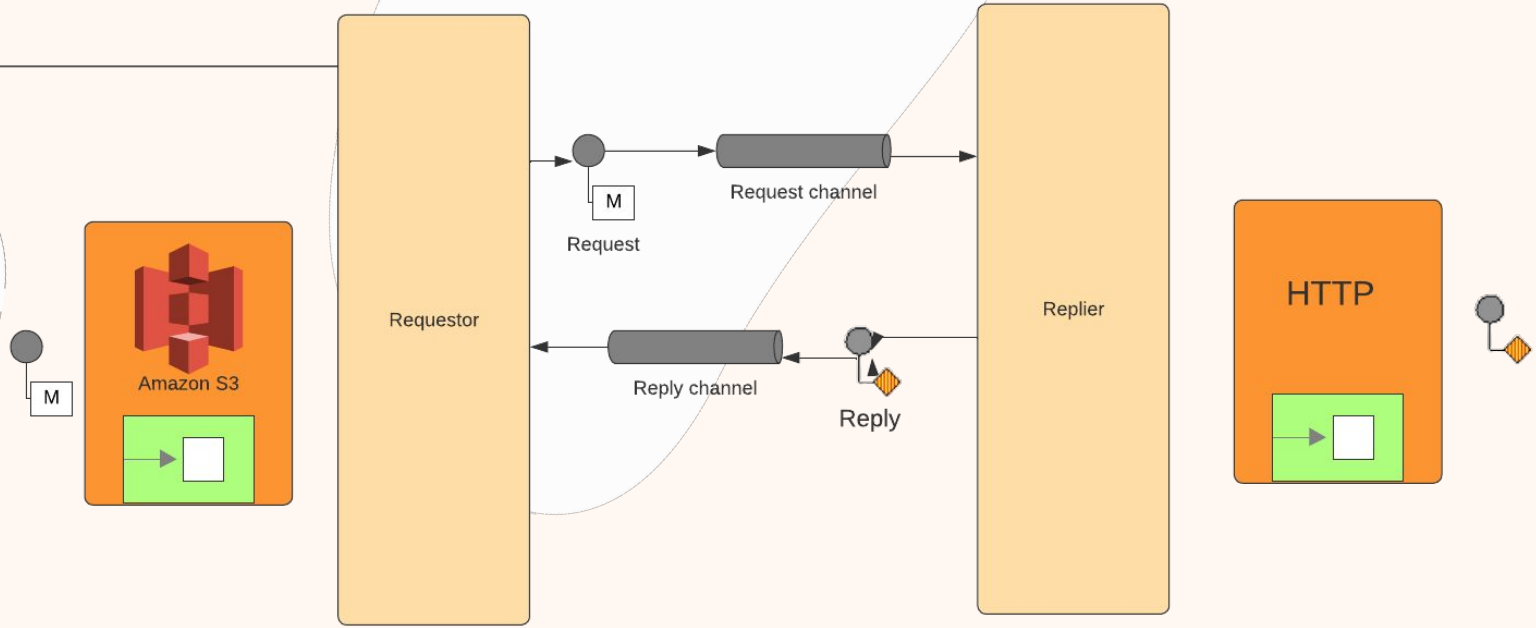
```
from("aws2-s3:bucketName")
    .to("http:my-host/api/path")
```

Libraries
POJO
Biz Logic...

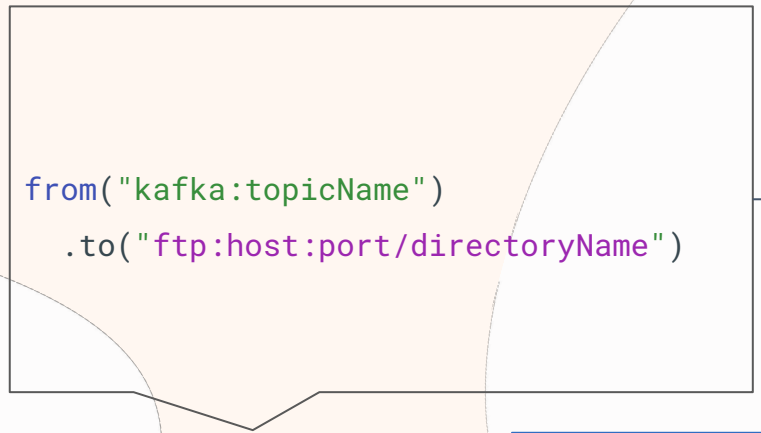


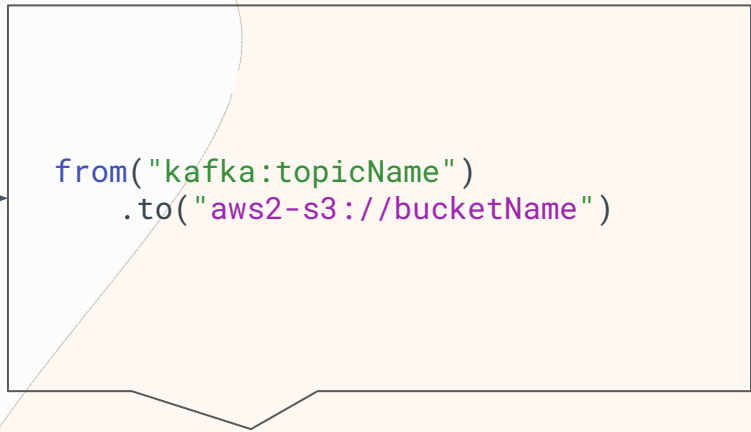
Apache Camel: Domain Specific Language (DSL)

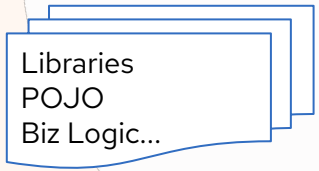
```
from("aws2-s3:bucketName")
    .to("http:my-host/api/path")
```

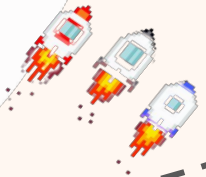


Apache Camel: Domain Specific Language (DSL)



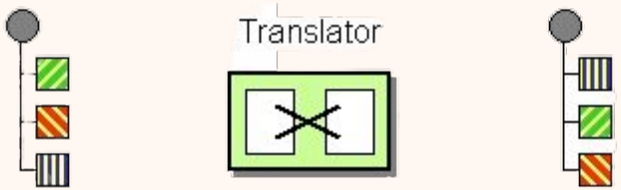




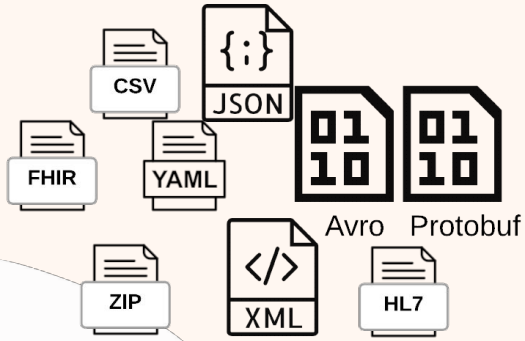


2

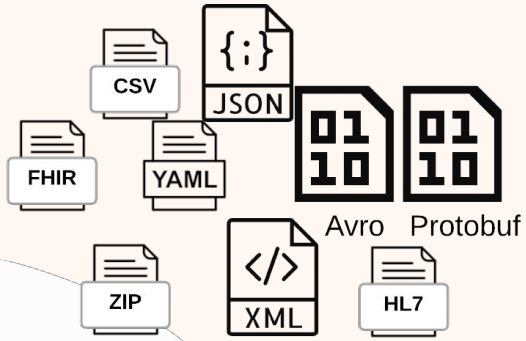
Data Transformation



Using Apache Camel Data formats



Using Apache Camel Data formats



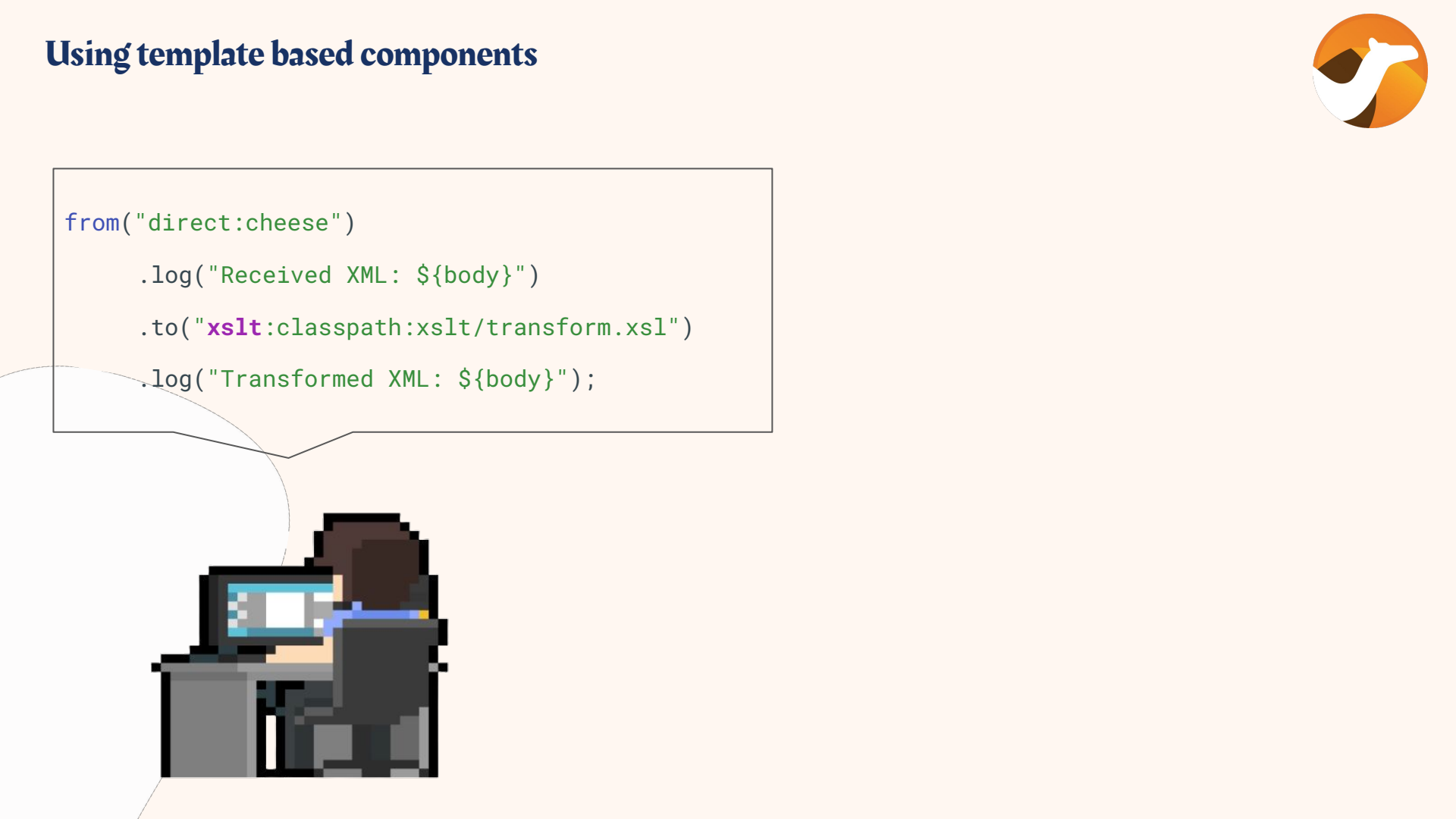

```
from("kafka:topic")
    .unmarshal().json()
    .to("http:my-host/api/path");
```

Using Translator or Set Body

```
from("direct:cheese")
      .setBody(simple("Hello ${body}"))
      .to("log:hello");
```

```
from("direct:cheese")
      .transform(new DataType("myDataType"))
      .to("log:hello");
```

Using template based components



```
from("direct:cheese")
      .log("Received XML: ${body}")
      .to("xslt:classpath:xslt/transform.xsl")
      .log("Transformed XML: ${body}");
```

Using Processor

```
from("activemq:myQueue")
     .process(new Processor() {
          public void process(Exchange exchange) throws Exception{
               String payload = exchange.getIn().getBody(String.class);
               // do something with the payload and/or exchange here
               exchange.getIn().setBody("Changed body");
          }
     })
     .to("activemq:myOtherQueue");
```

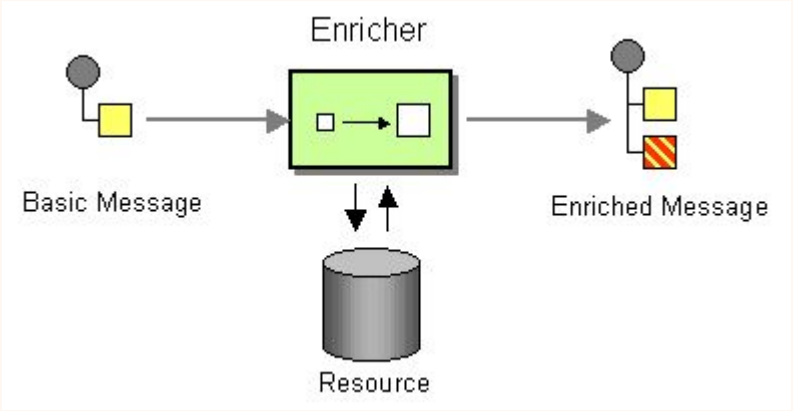
Using Beans

```
from("direct:hello")
.to("bean:com.foo.MyBean");
```

```
package com.foo;

public class MyBean {

      public String saySomething(String input){

            return "Hello " + input;

      }

}
```

Content Enricher

```
from("seda:a")
      .to("direct:myEnrichEndpoint")
      .to("seda:b");
```

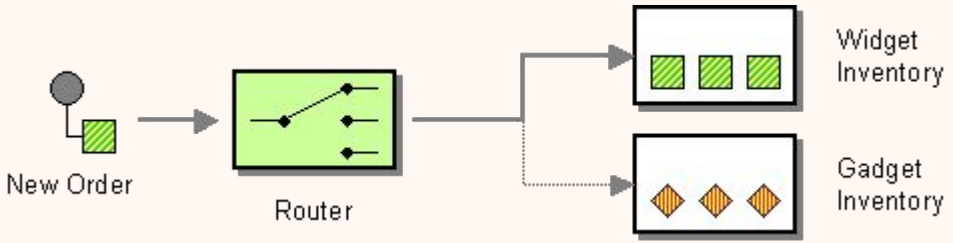


3

Routing Messages

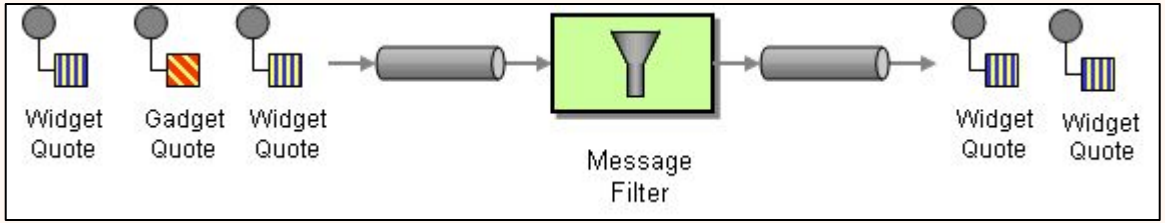
Content Based Router

```
from("seda:a").choice()
      .when(header("foo").isEqualTo("bar")).to("seda:b")
      .when(header("foo").isEqualTo("cheese")).to("seda:c")
      .otherwise().to("seda:d");
```



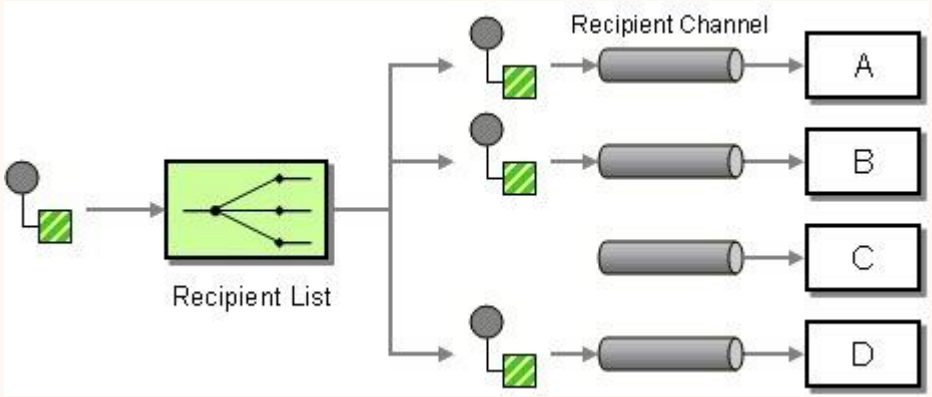
Message Filter

```
from("seda:a")
      .filter(header("foo").isEqualTo("bar")).to("seda:b");
```



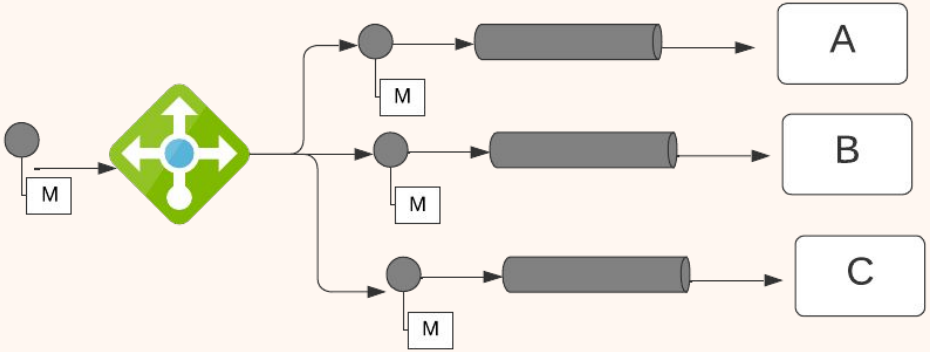
Recipient List

```
from("seda:a")
      .to("seda:b", "seda:c", "seda:d");
```

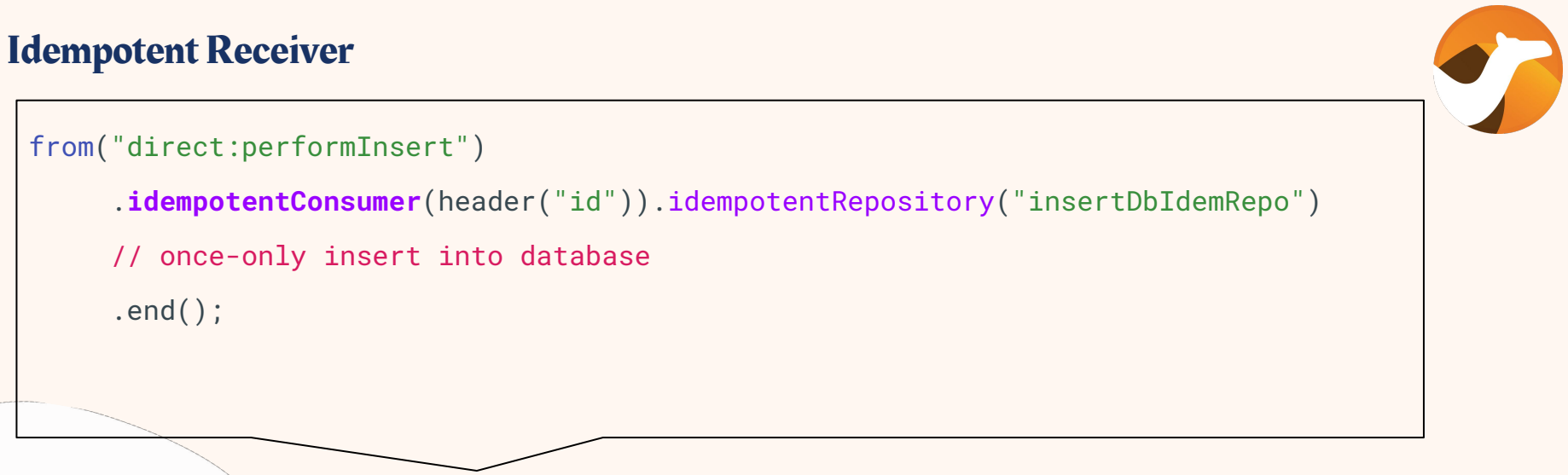


Load Balancer

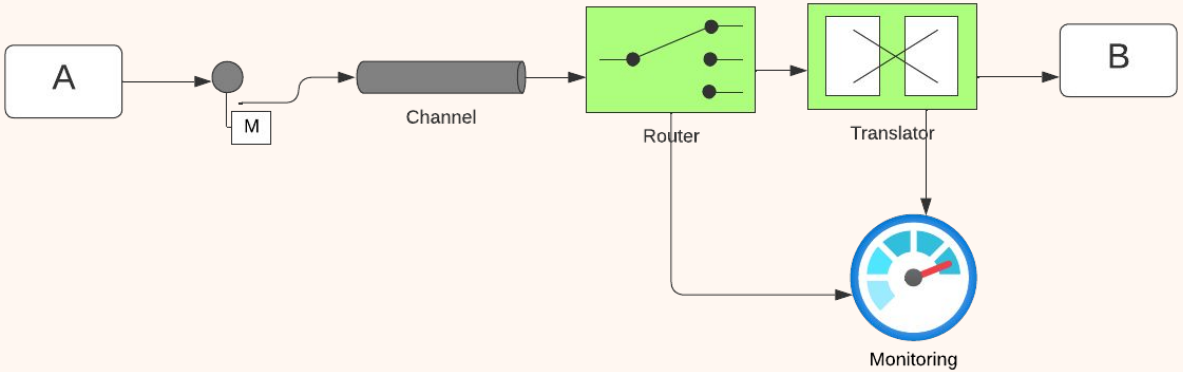
```
from("direct:start")
      .loadBalance().roundRobin()
      .to("mock:x", "mock:y", "mock:z");
```



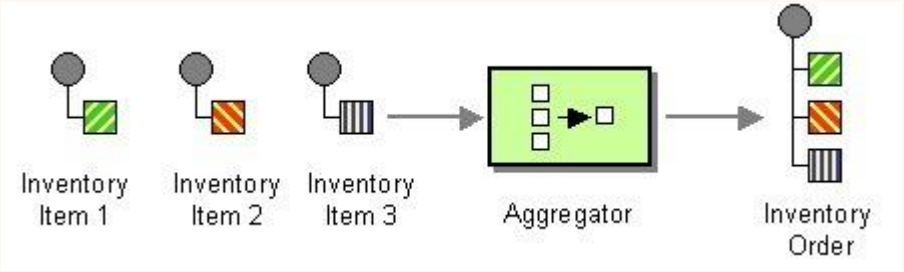
Idempotent Receiver

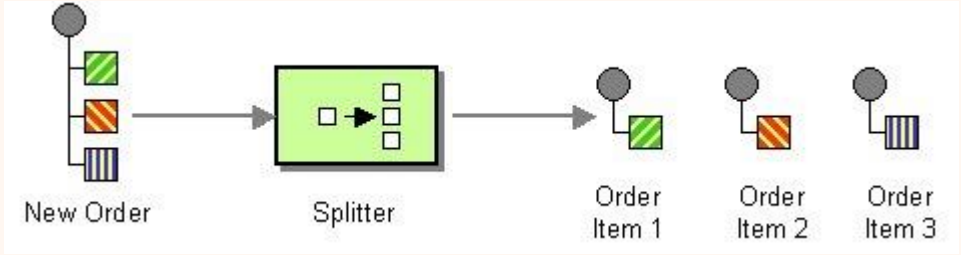


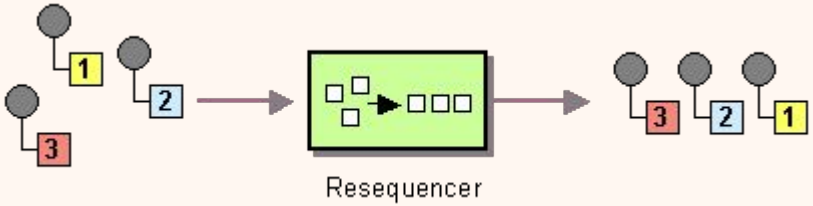
```
from("direct:performInsert")
      .idempotentConsumer(header("id")).idempotentRepository("insertDbIdemRepo")
      // once-only insert into database
      .end();
```



Using others EIPs

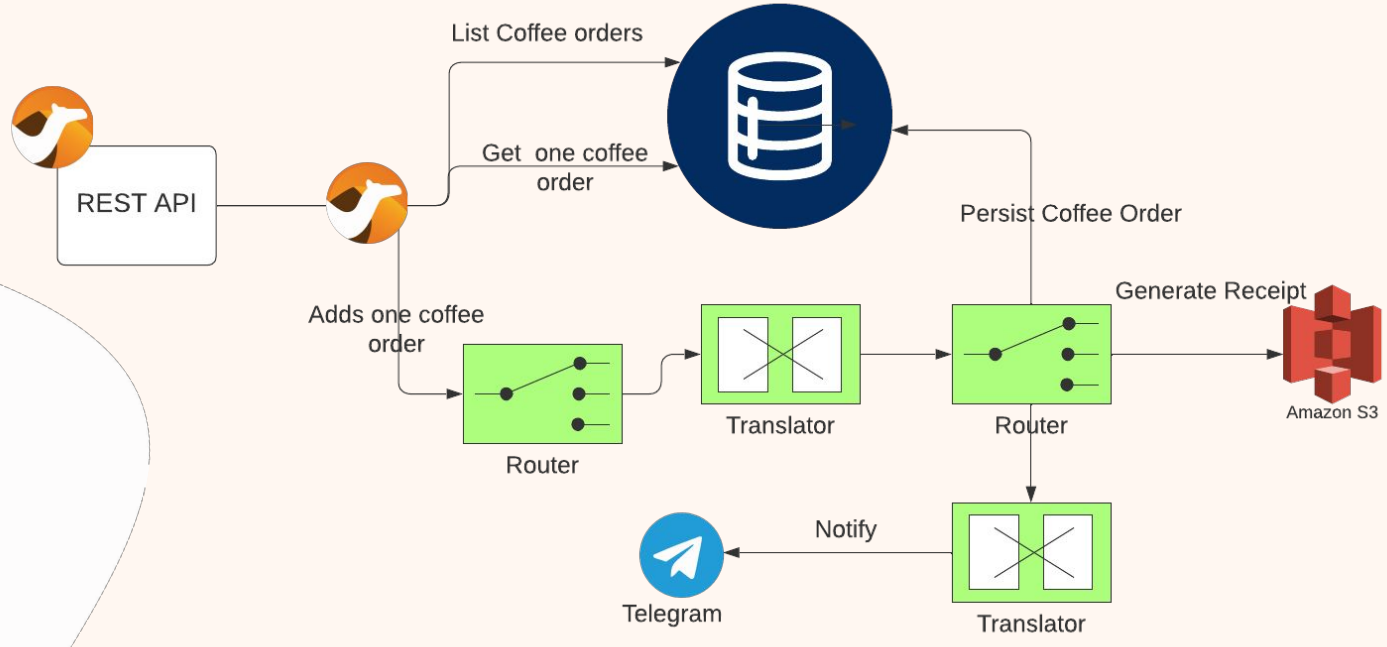






Demo #1

Demo #1


List Coffee orders
REST API
Get one coffee order
Persist Coffee Order
Generate Receipt
Adds one coffee order
Translator
Router
Amazon S3
Router
Notify
Telegram
Translator

Additional Challenges

4

Logging

Logging

- Logging Components
- Log EIP
- Customizing logs and format
- Masking sensitive information

```
from("activemq:orders")
      .to("log:com.mycompany.order?level=DEBUG&groupSize=10")
      .to("bean:processOrder");
```

```
from("direct:start")
      .log("Processing ${id}")
      .to("bean:foo");
```

5

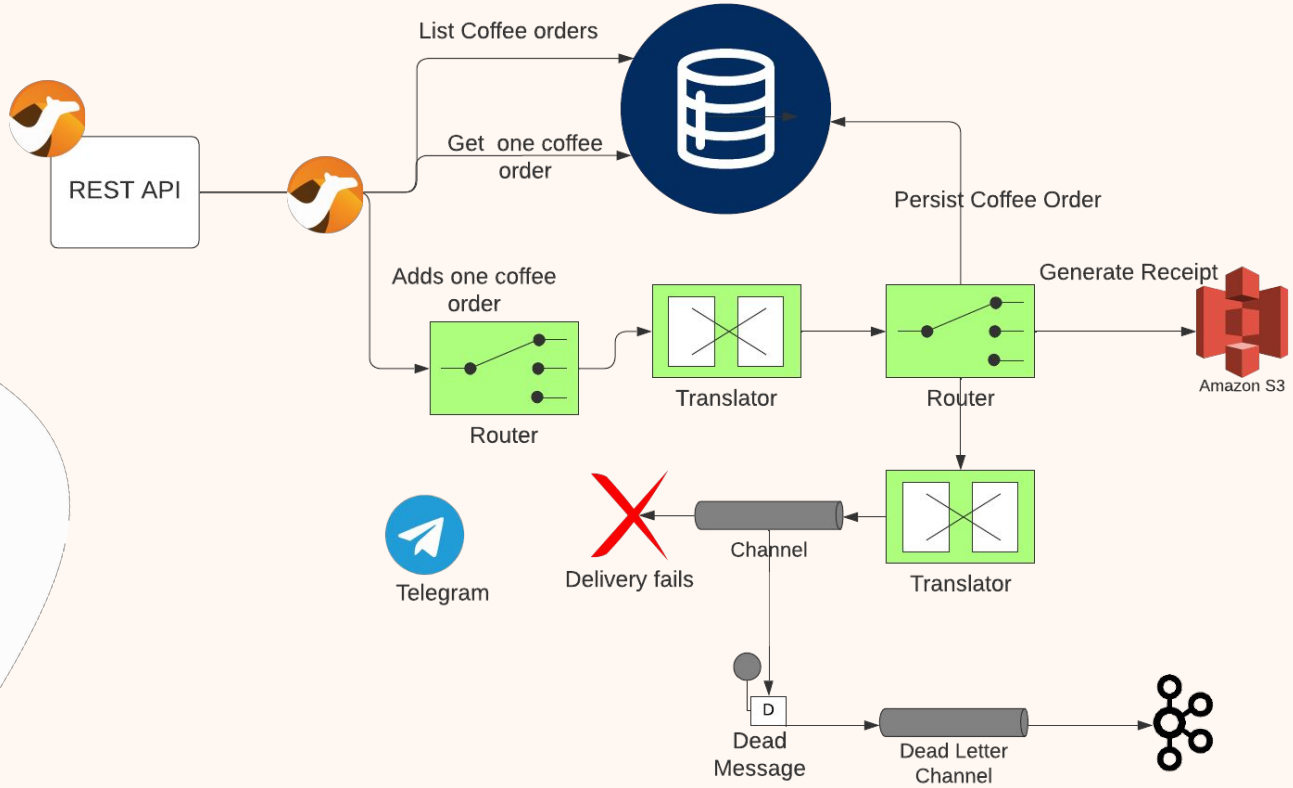
Error Handling

Error Handler: Dead Letter Queue

```
from("direct:start")
      .to("direct:invalidEndpoint"); // This will trigger an error

// Configure the Dead Letter Channel (DLC) to handle errors
errorHandler(deadLetterChannel("direct:errorQueue")
      .maximumRedeliveries(3) // Maximum number of redelivery attempts
      .redeliveryDelay(1000) // Delay between redelivery attempts
      .logExhausted(true) // Log if redelivery attempts are exhausted
);

// Define the DLC route to handle failed messages
from("direct:errorQueue")
…;
```

Demo #2

Demo #2


List Coffee orders
REST API
Get one coffee order
Persist Coffee Order
Adds one coffee order
Generate Receipt
Amazon S3
Translator
Router
Router
Telegram
Delivery fails
Channel
Translator
D
Dead Message
Dead Letter Channel

Danke
Thank You