



COEN 6731: Distributed Software Systems

Assignment – III

Submitted to – YAN LIU

Submitted by –Rohan K (40196377)

Date of submission – April 18, 2023

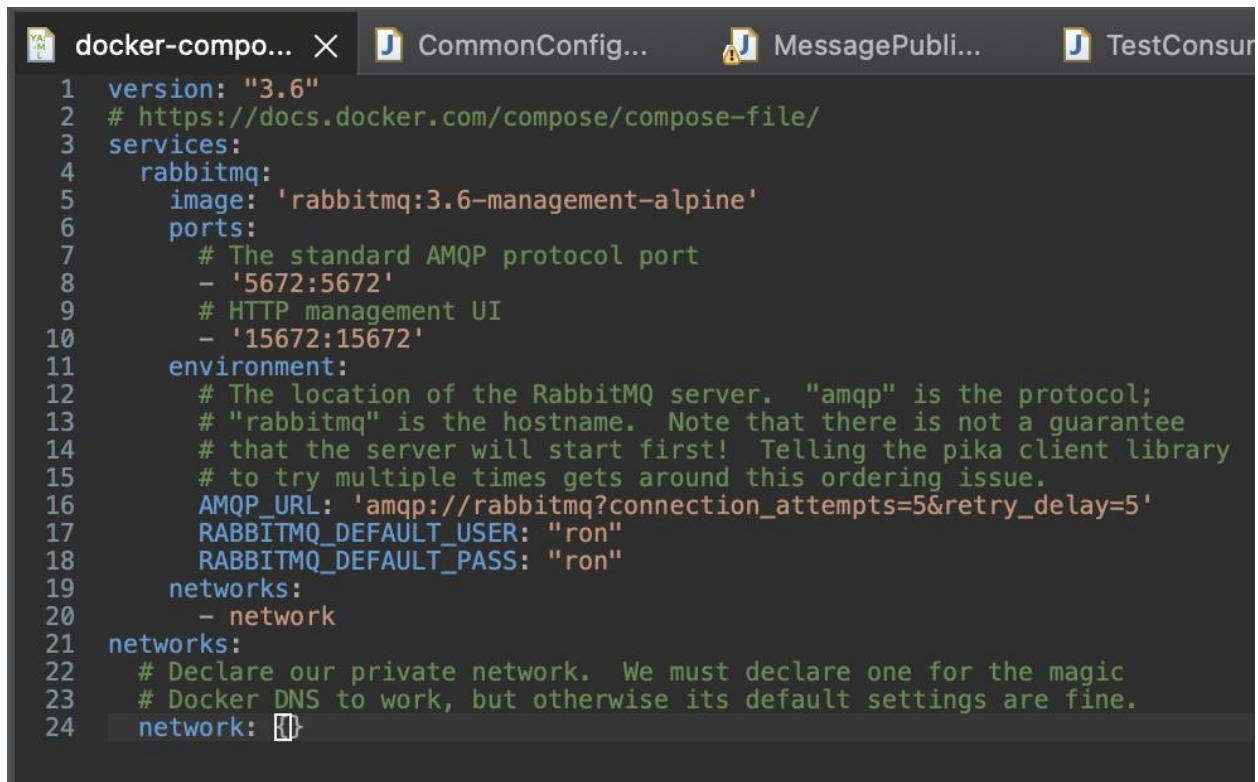
Task 1. Install rabbitmq server

For this assignment I have set up a RabbitMQ server for the development purpose using docker-compose. I have downloaded the official rabbitmq docker image to run it.

Prerequisites for this was to have docker and docker-compose installed by running the below commands:

```
$ docker -v and $ docker-compose -v
```

Further I have created a docker-compose.yml file with the below content –



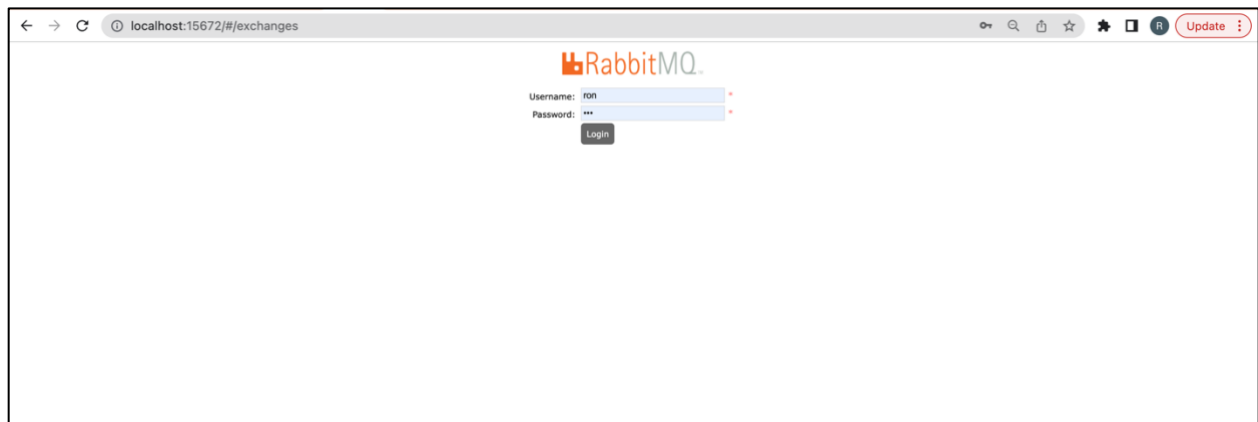
```
1 version: "3.6"
2 # https://docs.docker.com/compose/compose-file/
3 services:
4   rabbitmq:
5     image: 'rabbitmq:3.6-management-alpine'
6     ports:
7       # The standard AMQP protocol port
8       - '5672:5672'
9       # HTTP management UI
10      - '15672:15672'
11   environment:
12     # The location of the RabbitMQ server. "amqp" is the protocol;
13     # "rabbitmq" is the hostname. Note that there is not a guarantee
14     # that the server will start first! Telling the pika client library
15     # to try multiple times gets around this ordering issue.
16     AMQP_URL: 'amqp://rabbitmq?connection_attempts=5&retry_delay=5'
17     RABBITMQ_DEFAULT_USER: "ron"
18     RABBITMQ_DEFAULT_PASS: "ron"
19   networks:
20     - network
21 networks:
22   # Declare our private network. We must declare one for the magic
23   # Docker DNS to work, but otherwise its default settings are fine.
24   network: {}
```

As seen, the rabbitmq user name and password is – “ron”

Then I ran the docker-compose up command in a terminal, waited for the command to complete its execution. This downloaded the specified rabbitmq docker image, ran the rabbitmq-admin UI on #15672 and the Server listens on #5672 for messaging. Once the server is up, below log is seen -

```
~/Users/rohankodavalla/Documents/rabbitmq_test -- docker-compose up  ~/Documents/rabbitmq_test -- -bash  ~/Downloads -- opc@final-instance:~ -- -bash +
(base) rohans-air:rabbitmq_test rohankodavalla$ docker-compose up
[*] Running 1/0
  Container rabbitmq_test-rabbitmq-1 Created
Attaching to rabbitmq_test-rabbitmq-1
rabbitmq_test-rabbitmq-1
rabbitmq_test-rabbitmq-1 | =INFO REPORT==== 17-Apr-2023:21:24:05 ===
rabbitmq_test-rabbitmq-1 | Starting RabbitMQ 3.6.16 on Erlang 20.3.4
rabbitmq_test-rabbitmq-1 | Copyright (C) 2007-2018 Pivotal Software, Inc.
rabbitmq_test-rabbitmq-1 | Licensed under the MPL. See http://www.rabbitmq.com/
rabbitmq_test-rabbitmq-1 |
rabbitmq_test-rabbitmq-1 |           RabbitMQ 3.6.16. Copyright (C) 2007-2018 Pivotal Software, Inc.
rabbitmq_test-rabbitmq-1 |           Licensed under the MPL. See http://www.rabbitmq.com/
rabbitmq_test-rabbitmq-1 |
rabbitmq_test-rabbitmq-1 | ##### Logs: tty
rabbitmq_test-rabbitmq-1 | #####      tty
rabbitmq_test-rabbitmq-1 | #####
rabbitmq_test-rabbitmq-1 | Starting broker...
rabbitmq_test-rabbitmq-1 |
rabbitmq_test-rabbitmq-1 | =INFO REPORT==== 17-Apr-2023:21:24:05 ===
rabbitmq_test-rabbitmq-1 | node       : rabbit@ab22b3ab02a4
rabbitmq_test-rabbitmq-1 | home dir   : /var/lib/rabbitmq
rabbitmq_test-rabbitmq-1 | config file(s) : /etc/rabbitmq/rabbitmq.config
rabbitmq_test-rabbitmq-1 | cookie hash : 6nF9cUeJulDa0AsR8rfxYg==
rabbitmq_test-rabbitmq-1 | log        : tty
rabbitmq_test-rabbitmq-1 | sasl log    : tty
rabbitmq_test-rabbitmq-1 | database dir : /var/lib/rabbitmq/mnesia/rabbit@ab22b3ab02a4
rabbitmq_test-rabbitmq-1 |
rabbitmq_test-rabbitmq-1 | =INFO REPORT==== 17-Apr-2023:21:24:06 ===
rabbitmq_test-rabbitmq-1 | Memory high watermark set to 1573 MiB (1649884982 bytes) of 3933 MiB (4124512256 bytes) total
rabbitmq_test-rabbitmq-1 |
rabbitmq_test-rabbitmq-1 | =INFO REPORT==== 17-Apr-2023:21:24:06 ===
rabbitmq_test-rabbitmq-1 | Enabling free disk space monitoring
rabbitmq_test-rabbitmq-1 |
rabbitmq_test-rabbitmq-1 | =INFO REPORT==== 17-Apr-2023:21:24:06 ===
rabbitmq_test-rabbitmq-1 | Disk free limit set to 50MB
rabbitmq_test-rabbitmq-1 |
rabbitmq_test-rabbitmq-1 | =INFO REPORT==== 17-Apr-2023:21:24:06 ===
rabbitmq_test-rabbitmq-1 | Limiting to approx 1848476 file handles (943626 sockets)
rabbitmq_test-rabbitmq-1 |
rabbitmq_test-rabbitmq-1 | =INFO REPORT==== 17-Apr-2023:21:24:06 ===
rabbitmq_test-rabbitmq-1 | FHC read buffering: OFF
rabbitmq_test-rabbitmq-1 | FHC write buffering: ON
rabbitmq_test-rabbitmq-1 |
rabbitmq_test-rabbitmq-1 | =INFO REPORT==== 17-Apr-2023:21:24:06 ===
rabbitmq_test-rabbitmq-1 | Waiting for Mnesia tables for 30000 ms, 9 retries left
rabbitmq_test-rabbitmq-1 |
rabbitmq_test-rabbitmq-1 | =INFO REPORT==== 17-Apr-2023:21:24:06 ===
rabbitmq_test-rabbitmq-1 | Waiting for Mnesia tables for 30000 ms, 9 retries left
rabbitmq_test-rabbitmq-1 |
rabbitmq_test-rabbitmq-1 | =INFO REPORT==== 17-Apr-2023:21:24:06 ===
rabbitmq_test-rabbitmq-1 | Priority queues enabled, real BQ is rabbit_variable_queue
rabbitmq_test-rabbitmq-1 |
rabbitmq_test-rabbitmq-1 | =INFO REPORT==== 17-Apr-2023:21:24:06 ===
rabbitmq_test-rabbitmq-1 | Starting rabbit_node_monitor
rabbitmq_test-rabbitmq-1 |
rabbitmq_test-rabbitmq-1 | =INFO REPORT==== 17-Apr-2023:21:24:06 ===
rabbitmq_test-rabbitmq-1 | Management plugin: using rates mode 'basic'
rabbitmq_test-rabbitmq-1 |
rabbitmq_test-rabbitmq-1 | =INFO REPORT==== 17-Apr-2023:21:24:06 ===
rabbitmq_test-rabbitmq-1 | msg_store_transient: using rabbit_msg_store_ets_index to provide index
rabbitmq_test-rabbitmq-1 |
```

Further, when navigated to <http://localhost:15672> in browser and logged in to the management dashboard with “ron” as both username and password, below is the RabbitMQ admin management dashboard observed –



The UI when logged in is as below -

The screenshot shows the RabbitMQ 3.6.16 web interface running on Erlang 20.3.4. The browser address bar shows `localhost:15672/#/exchanges`. The interface includes a navigation bar with tabs for Overview, Connections, Channels, Exchanges (selected), Queues, and Admin. The Exchanges tab displays a list of 10 exchanges. Below the list is a form to 'Add a new exchange' with fields for Name, Type (fanout), Durability (Durable), Auto delete (No), Internal (No), and Arguments. The footer contains links for HTTP API, Server Docs, Tutorials, Community Support, Community Slack, Commercial Support, Plugins, GitHub, and Changelog.

Exchanges

▼ All exchanges (10)

Pagination: Page 1 of 1 - Filter: ☐ Regex [?](#) Displaying 10 items, page size up to: 100

Name	Type	Features	Message rate in	Message rate out	+/-
(AMQP default)	direct	D			
EduCostStatExchange	topic	D	0.00/s	0.00/s	
Exchange	topic	D			
amq.direct	direct	D			
amq.fanout	fanout	D			
amq.headers	headers	D			
amq.match	headers	D			
amq.rabbitmq.log	topic	D I			
amq.rabbitmq.trace	topic	D I			
amq.topic	topic	D			

▼ Add a new exchange

Name:

Type: [fanout](#)

Durability: [Durable](#)

Auto delete: [No](#)

Internal: [No](#)

Arguments: = [String](#)

[Add](#) [Alternate exchange](#) [?](#)

[Add exchange](#)

HTTP API | Server Docs | Tutorials | Community Support | Community Slack | Commercial Support | Plugins | GitHub | Changelog

So, the RabbitMQ installation is successful.

Task 2. Program the producer and consumer using rabbitmq exchange topic libraries

In order to Send and Receive a message in RabbitMQ below steps were followed -

1) Programmed the Publisher :

```
docker-compo... MessagePubli... CreateQueues... rabbitmq-pa... »19

90 package com.amqp.basic.queue;
91
92
93 import com.mongodb.client.FindIterable;
94 import com.mongodb.client.MongoClient;
95 import com.mongodb.client.MongoClients;
96 import com.mongodb.client.MongoCollection;
97 import com.mongodb.client.MongoDatabase;
98 import com.mongodb.client.model.Filters;
99 import com.mongodb.client.model.Sorts;
100
101 import org.bson.Document;
102
103 import com.rabbitmq.client.Channel;
104 import com.rabbitmq.client.Connection;
105 import com.rabbitmq.client.ConnectionFactory;
106 import com.rabbitmq.client.Consumer;
107 import com.rabbitmq.client.DefaultConsumer;
108
109 import java.io.IOException;
110 import java.util.concurrent.TimeoutException;
111
112 // Run me - third
113 public class MessagePublisher {
114     public static void main(String[] args) throws IOException, TimeoutException {
115         ConnectionFactory factory = new ConnectionFactory();
116         factory.setHost("localhost");
117         factory.setUsername("ron");
118         factory.setPassword("ron");
119         factory.setPort(5672);
120         factory.setVirtualHost("/");
121
122
123         try (Connection connection = factory.newConnection();
124             Channel channel = connection.createChannel()) {
125
126             // Declare the exchanges
127             channel.exchangeDeclare("EduCostStatExchange", "topic", true);
128             //channel.exchangeDeclare("EduCostStatExchange", "topic", true);
129             //channel.exchangeDeclare("EduCostStatQueryThree", "topic3");
130             //channel.exchangeDeclare("EduCostStatQueryFour", "topic4");
131             //channel.exchangeDeclare("EduCostStatQueryFive", "topic5");
132
133             Consumer consumer = new DefaultConsumer(channel) {
134                 //...
135             };
136
137
138
139             // MongoDB database and collection setup
140             MongoClient mongoClient = MongoClients.create("mongodb+srv://rkodava:Dimpu1997@educoststat.ioim58e.mongodb.net/?retryWrites=true");
141             MongoDatabase database = mongoClient.getDatabase("test");
142             MongoCollection<Document> collection = database.getCollection("EduCostStatQueryFive");
143             FindIterable<Document> results = collection.find();
144
145
146             // Query for EduCostStatQueryOne
147             /*
148             int year = 2013;
149             String state = "Alabama";
150             String type = "Private";
151             String length = "4-year";
152             String expense = "Fees/Tuition";
153             String message = "Message for EduCostStatQueryOne";
154
155             // Query for EduCostStatQueryTwo
156             /*
157             int year = 2013;
158             String type = "Private";
159             String length = "4-year";
160             String message = "Message for EduCostStatQueryTwo";
161
162             // Query for EduCostStatQueryThree
```

```

138
139 // MongoDB database and collection setup
140 MongoClient mongoClient = MongoClient.create("mongodb+srv://rkodava:Dimpu1997@educoststat.ioim58e.mongodb.net/?retryWrites
141 MongoDB database = mongoClient.getDatabase("test");
142 MongoCollection<Document> collection = database.getCollection("EduCostStatQueryFive");
143 FindIterable<Document> results = collection.find();
144
145
146 // Query for EduCostStatQueryOne
147 /*
148 int year = 2013;
149 String state = "Alabama";
150 String type = "Private";
151 String length = "4-year";
152 String expense = "Fees/Tuition";
153 String message = "Message for EduCostStatQueryOne";
154
155 // Query for EduCostStatQueryTwo
156 /*
157 int year = 2013;
158 String type = "Private";
159 String length = "4-year";
160 String message = "Message for EduCostStatQueryTwo";
161
162 // Query for EduCostStatQueryThree
163
164 int year = 2013;
165 String type = "Private";
166 String length = "4-year";
167 String message = "Message for EduCostStatQueryThree";
168
169 // Query for EduCostStatQueryFour
170 int latestYear = 2021;
171 String type = "Public";
172 String length = "4-year";
173 int[] pastYears = new int[]{2020, 2019, 2018, 2017};
174 String message = "Message for EduCostStatQueryFour";
175 */
176 // Query for EduCostStatQueryFive
177
178 int year = 2019;
179 String type = "Public In-State";
180 String length = "4-year";
181 String message = "Message for EduCostStatQueryFive";
182
183 // Display query results
184 for (Document result : results) {
185     System.out.println(result.toJson());
186 }
187 // Publish the message to the exchange for EduCostStatQueryOne
188 //channel.basicPublish("EduCostStatExchange", "Cost-" + year + "-" + state + "-" + type + "-" + length, null, message.getBytes()
189
190 // Publish the message to the exchange for EduCostStatQueryTwo
191 //message = "Message for EduCostStatQueryTwo";
192 //channel.basicPublish("EduCostStatExchange", "Top5-Expensive-" + year + "-" + type + "-" + length, null, message.getBytes()
193
194 // Publish the message to the exchange for EduCostStatQueryThree
195 // message = "Message for EduCostStatQueryThree";
196 // channel.basicPublish("EduCostStatExchange", "Top5-Economic-" + year + "-" + type + "-" + length, null, message.getBytes()
197
198 // Publish the message to the exchange for EduCostStatQueryFour
199 // message = "Message for EduCostStatQueryFour";
200 //channel.basicPublish("EduCostStatExchange", "Top5-HighestGrow-2020-2022-5-" + type + "-" + length, null, message.getBytes()
201 //channel.basicPublish("EduCostStatExchange", "Top5-HighestGrow-2021" + latestYear+ "-" + type + "-" + length + "-" + pastYe
202 // Publish the message to the exchange for EduCostStatQueryFive
203 message = "Message for EduCostStatQueryFive";
204 channel.basicPublish("EduCostStatExchange", "AverageExpense-" + year + "-" + type + "-" + length, null, message.getBytes());
205
206 }
207 }
208 }
209
210

```

The above is a Java code for publishing a message to a RabbitMQ exchange based on data retrieved from a MongoDB database. Here's a breakdown of the code:

- The necessary libraries are imported for MongoDB and RabbitMQ.
- The main method is defined, which throws IOException and TimeoutException exceptions.
- A new ConnectionFactory object is created, which specifies the connection parameters for RabbitMQ such as the hostname, username, password, port, and virtual host.
- A new connection and channel to RabbitMQ are created and declared.
- A new Consumer object is created for the RabbitMQ channel.

- A new MongoClient object is created, which specifies the connection parameters for MongoDB such as the connection string.
- The database and collection are specified and retrieved using the MongoDB and MongoCollection objects.
- A MongoDB query is executed to retrieve the data from the collection specified.
- Depending on the query, the message is created with a message for a particular query result.

The message is published to the RabbitMQ exchange based on the query using the basicPublish method of the channel.

In this code snippet it is commented to indicate which part of the code corresponds to each query. However, only the code for the last query (EduCostStatQueryFive) is executed since the code for the other queries is commented out.

2) Programmed the Subscriber / Consumer :

```

MessageSubscriber.java X
28
29 import com.rabbitmq.client.*;
30
31 import java.io.IOException;
32 import java.util.concurrent.TimeoutException;
33
34 // Run me - fourth
35 public class MessageSubscriber {
36     public static void main(String[] args) throws IOException, TimeoutException {
37         ConnectionFactory factory = new ConnectionFactory();
38         factory.setHost("localhost");
39         factory.setUsername("ron");
40         factory.setPassword("ron");
41         factory.setPort(5672);
42         factory.setVirtualHost("/");
43
44         String exchangeName = "EduCostStatExchange";
45         //String queueName = "topic1";
46         //String topic = "Cost-2013-Alabama-Private-4-year";
47         //String queueName = "topic2";
48         //String topic = "Top5-Expensive-2013-Private-4-year";
49         //String queueName = "topic3";
50         //String topic = "Top5-Economic-2013-Private-4-year";
51         //String queueName = "topic4";
52         //String topic = "Top5-HighestGrow-2021";
53         String queueName = "topic5";
54         String topic = "AverageExpense-2019-Public In-State-4-year";
55
56
57         try (Connection connection = factory.newConnection();
58             Channel channel = connection.createChannel()) {
59
60             // Declare the exchange
61             channel.exchangeDeclare(exchangeName, "topic", true);
62
63             // Declare the queue
64             channel.queueDeclare(queueName, true, false, false, null);
65
66             // Bind the queue to the exchange
67             channel.queueBind(queueName, exchangeName, topic);
68
69             // Create a consumer
70             DeliverCallback deliverCallback = (consumerTag, delivery) -> {
71                 String message = new String(delivery.getBody(), "UTF-8");
72                 //System.out.println(" [x] Received '" + delivery.getEnvelope().getRoutingKey()+"': '"+message + "'");
73                 //System.out.println("Message received and processed successfully.");
74
75             };
76             channel.basicConsume(queueName, true, deliverCallback, consumerTag -> {});
77             System.out.println("Waiting for messages...");
78
79             /*Consumer consumer = new DefaultConsumer(channel) {
80                 @Override
81                 public void handleDelivery(String consumerTag, Envelope envelope, AMQP.BasicProperties properties, byte
82                     String message = new String(body, "UTF-8");
83                     System.out.println("Received message: " + message);
84                 }
85             };
86
87             // Start consuming messages
88             channel.basicConsume(queueName, true, consumer);*/
89
90     }

```

The program establishes a connection to a RabbitMQ server using the `ConnectionFactory` class, creates a channel to communicate with the server, and declares an exchange and a queue. It then binds the queue to the exchange and consumes messages from the queue using a `DeliverCallback`.

Here is a summary of what the program does:

- Imports the necessary classes from the RabbitMQ Java client library, as well as `IOException` and `TimeoutException` from the standard Java library.
- Defines a class called `MessageSubscriber` with a main method that takes no arguments.
- Creates a `ConnectionFactory` object and sets the host, username, password, port, and virtual host to connect to a RabbitMQ server running on localhost.
- Defines an exchange name, a queue name, and a topic name.
- Establishes a connection to the RabbitMQ server and creates a channel for communication.
- Declares the exchange using the `exchangeDeclare` method of the Channel object, specifying the exchange name, type ("topic"), and durability (true).
- Declares the queue using the `queueDeclare` method of the Channel object, specifying the queue name, durability (true), `autoDelete` (false), `exclusive` (false), and additional arguments (null).
- Binds the queue to the exchange using the `queueBind` method of the Channel object, specifying the queue name, exchange name, and topic name.
- Defines a `DeliverCallback` that prints the message received from the queue to the console.
- Consumes messages from the queue using the `basicConsume` method of the Channel object, specifying the queue name, `autoAck` (true), the `DeliverCallback`, and a `CancelCallback` that does nothing.
- Prints "Waiting for messages..." to the console to indicate that the program is waiting for messages.

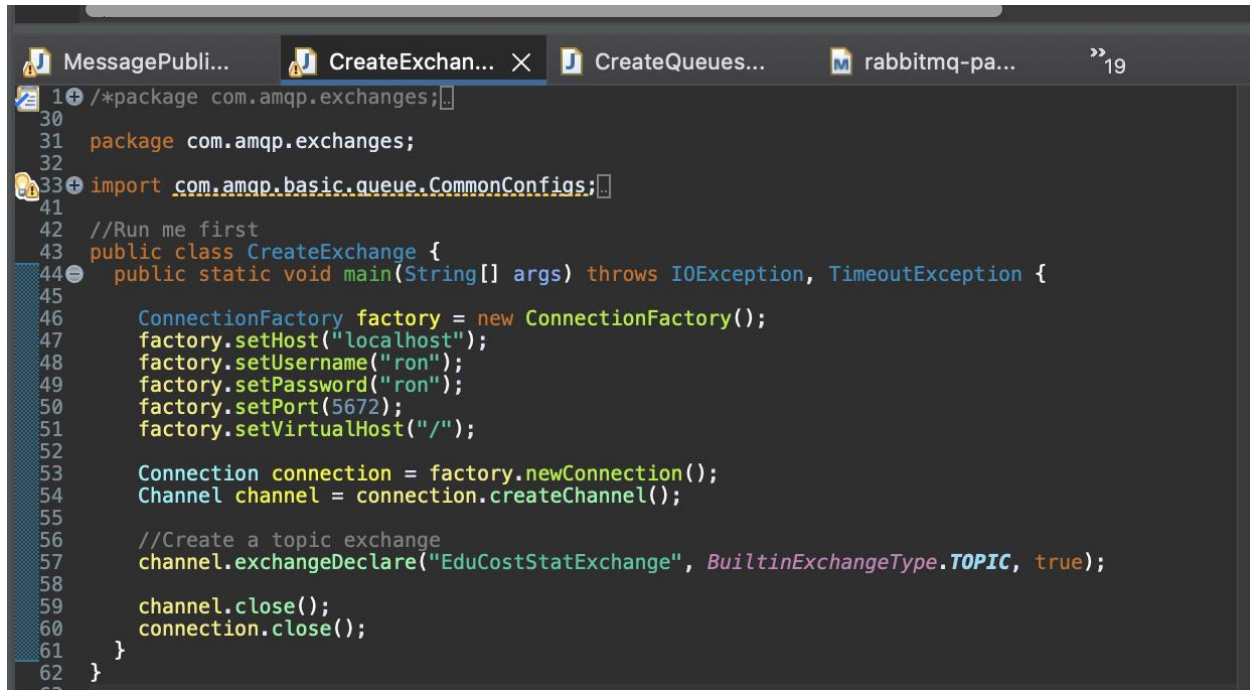
The commented out code shows an alternative way of consuming messages using a `DefaultConsumer` object, which implements the `Consumer` interface and defines a `handleDelivery` method that is called when a message is received.

Q 1, 2, 3 of assignment -

So, now we can confirm that the producer code retrieves the datasets from each collection from the MongoDB cloud service for each topics listed in the table above.

Now, it is required to program Exchange, Queue and Binding. Below is the demonstration –

Declare an Exchange

A screenshot of an IDE window titled 'CreateExchan...' showing Java code for creating a RabbitMQ exchange. The code is as follows:

```
1+ /*package com.amqp.exchanges;...
30
31 package com.amqp.exchanges;
32
33+ import com.amqp.basic.queue.CommonConfigs;
41
42 //Run me first
43 public class CreateExchange {
44- public static void main(String[] args) throws IOException, TimeoutException {
45
46     ConnectionFactory factory = new ConnectionFactory();
47     factory.setHost("localhost");
48     factory.setUsername("ron");
49     factory.setPassword("ron");
50     factory.setPort(5672);
51     factory.setVirtualHost("/");
52
53     Connection connection = factory.newConnection();
54     Channel channel = connection.createChannel();
55
56     //Create a topic exchange
57     channel.exchangeDeclare("EduCostStatExchange", BuiltinExchangeType.TOPIC, true);
58
59     channel.close();
60     connection.close();
61 }
62 }
```

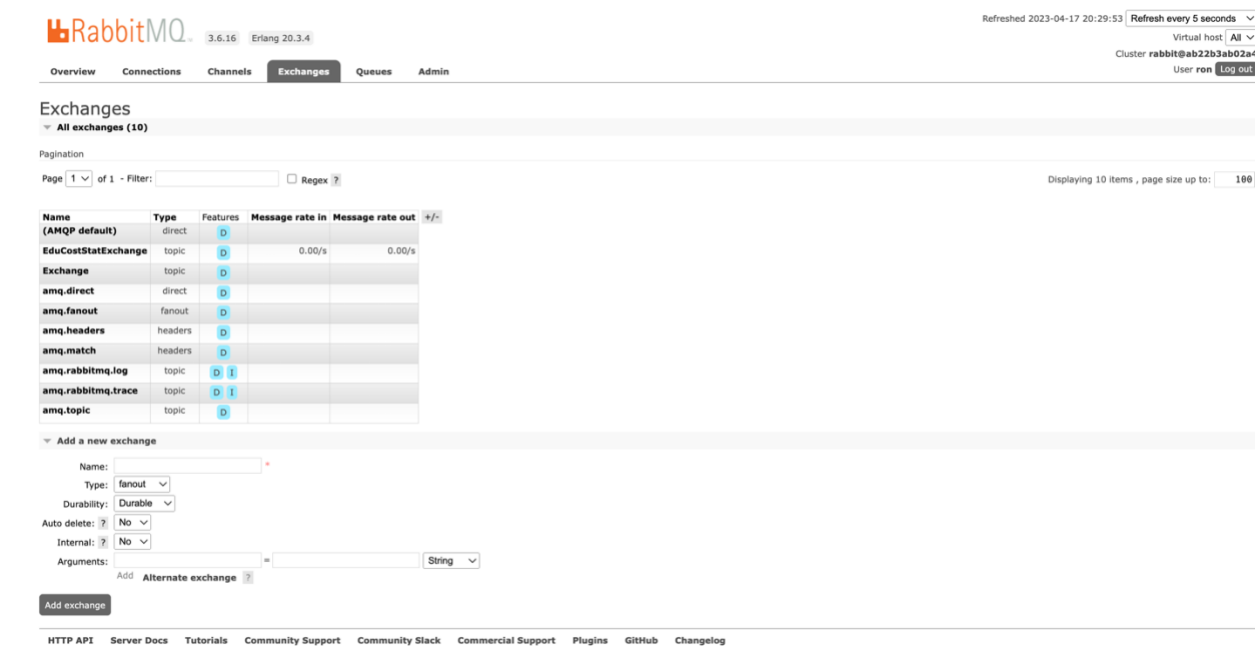
The above code is an example of how to create an exchange in RabbitMQ using the Java client library.

The ConnectionFactory is used to create a new connection to RabbitMQ, and the Connection is used to create a new channel for communication. The Channel is used to create the exchange using the exchangeDeclare() method.

As per the assignment architecture a topic exchange named **EduCostStatExchange** is created with BuiltinExchangeType.TOPIC. The boolean value true specifies that the exchange is durable, which means that it will survive a RabbitMQ broker restart. Finally, the Channel and Connection are closed to free up resources.

This code can be run as a standalone program to create the exchange on the RabbitMQ server. Once the exchange is created, producers can send messages to the exchange and consumers can consume messages from the exchange using the same exchange name.

The exchange – “EduCostStatExchange” can be verified on RabbitMQ server as shown below –



RabbitMQ 3.6.16 Erlang 20.3.4

Refreshed 2023-04-17 20:29:53 Refresh every 5 seconds

Virtual host All

Cluster rabbit@ab22b3ab02a4

User ron Log out

Overview Connections Channels **Exchanges** Queues Admin

Exchanges

▼ All exchanges (10)

Page 1 of 1 - Filter: ☐ Regex

Displaying 10 items, page size up to: 100

Name (AMQP default)	Type	Features	Message rate in	Message rate out	+/-
EduCostStatExchange	topic	D	0.00/s	0.00/s	
Exchange	topic	D			
amq.direct	direct	D			
amq.fanout	fanout	D			
amq.headers	headers	D			
amq.match	headers	D			
amq.rabbitmq.log	topic	D I			
amq.rabbitmq.trace	topic	D I			
amq.topic	topic	D			

▼ Add a new exchange

Name:

Type:

Durability:

Auto delete:

Internal:

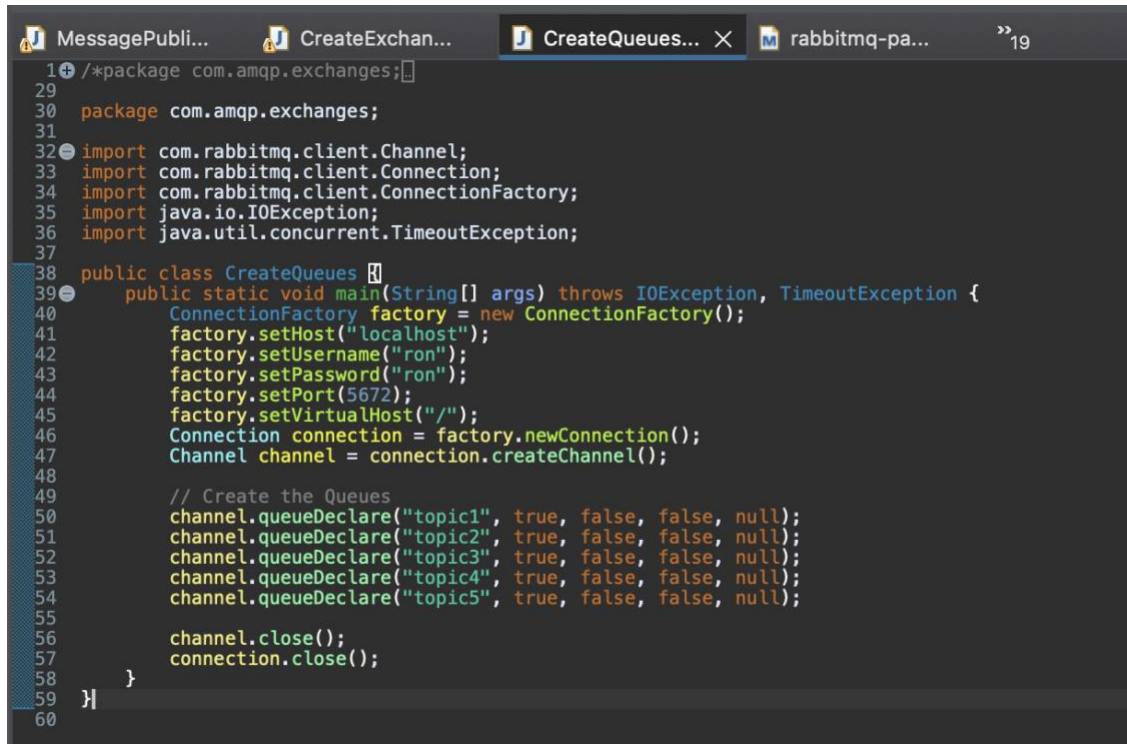
Arguments: = String

Add Alternate exchange

Add exchange

HTTP API Server Docs Tutorials Community Support Community Stack Commercial Support Plugins GitHub Changelog

Declare a Queue



```
1+ /*package com.amqp.exchanges;
29
30 package com.amqp.exchanges;
31
32 import com.rabbitmq.client.Channel;
33 import com.rabbitmq.client.Connection;
34 import com.rabbitmq.client.ConnectionFactory;
35 import java.io.IOException;
36 import java.util.concurrent.TimeoutException;
37
38 public class CreateQueues {
39     public static void main(String[] args) throws IOException, TimeoutException {
40         ConnectionFactory factory = new ConnectionFactory();
41         factory.setHost("localhost");
42         factory.setUsername("ron");
43         factory.setPassword("ron");
44         factory.setPort(5672);
45         factory.setVirtualHost("/");
46         Connection connection = factory.newConnection();
47         Channel channel = connection.createChannel();
48
49         // Create the Queues
50         channel.queueDeclare("topic1", true, false, false, null);
51         channel.queueDeclare("topic2", true, false, false, null);
52         channel.queueDeclare("topic3", true, false, false, null);
53         channel.queueDeclare("topic4", true, false, false, null);
54         channel.queueDeclare("topic5", true, false, false, null);
55
56         channel.close();
57         connection.close();
58     }
59 }
60
```

As per the assignment architecture, there are 5 queues declared, with names topic1, topic2, topic3, topic4, topic5 corresponding to 5 topics that are listed –

Collection	Parameters	Topic (* cannot change)
EduCostStatQueryOne	Query the cost given specific year, state, type, length, expense	Cost-[Year]-[State]-[Type]-[Length]
EduCostStatQueryTwo	Query the top 5 most expensive states (with overall expense) given a year, type, length	Top5-Expensive-[Year]-[Type]-[Length]
EduCostStatQueryThree	Query the top 5 most economic states (with overall expense) given a year, type, length	Top5-Economic-[Year]-[Type]-[Length]
EduCostStatQueryFour	Query the top 5 states of the highest growth rate of overall expense given a range of past years, one year, three years and five years (using the latest year as the base) , type and length	Top5-HighestGrow-[Years]
EduCostStatQueryFive	Aggregate region's average overall expense for a given year, type and length	AverageExpense-[Year]-[Type]-[Length]

The queues can be verified on RabbitMQ server as shown below –

RabbitMQ 3.6.16 Erlang 20.3.4 Refreshed 2023-04-17 23:25:25 Refresh every 5 seconds Virtual host: All Cluster: rabbit@ab22b3ab02a4 User: ron Log out

Overview Connections Channels Exchanges **Queues** Admin

Queues

↳ All queues (5)

Name	Features	State	Messages			Message rates		
			Ready	Unacked	Total	incoming	deliver	get / ack
topic1		idle	1	0	1	0.00/s	0.00/s	0.00/s
topic2		idle	1	0	1	0.00/s	0.00/s	0.00/s
topic3		idle	0	0	0			
topic4		idle	0	0	0			
topic5		idle	0	0	0			

▼ Add a new queue

Name:

Durability:

Auto delete:

Arguments: =

Add Message TTL | Auto expire | Max length | Max length bytes | Dead letter exchange | Dead letter routing key | Maximum priority | Lazy mode | Master locator

HTTP API Server Docs Tutorials Community Support Community Slack Commercial Support Plugins GitHub Changelog

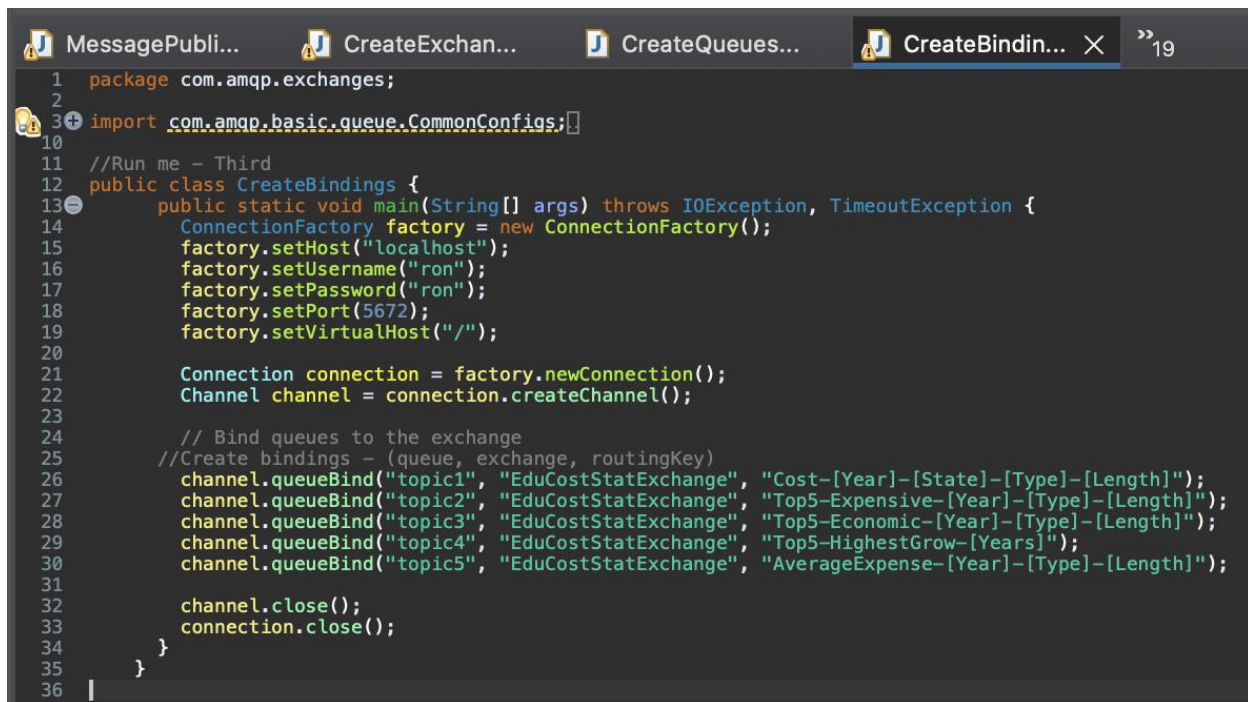
Binding a Queue with an Exchange

Bindings are an essential component of message routing in RabbitMQ. They serve as a means to establish a connection between Exchanges and Queues. It's important to note that messages are never directly published to a Queue in RabbitMQ.

When a producer sends a message to RabbitMQ, it is first routed to an exchange. Exchanges are responsible for routing messages and exist within a virtual host (vhost) in the RabbitMQ Server. They accept messages from the producer application and route them to message queues with the help of header attributes, bindings, and routing keys.

Bindings are essentially links that are set up between a queue and an exchange. They are responsible for establishing the connection between the two entities. Routing keys are message attributes that are used by the exchange to determine which queue to route the message to, based on the exchange type.

Depending on the exchange type, the exchange might examine the routing key of the message and use it to route the message to the appropriate queue.



```
1 package com.amqp.exchanges;
2
3 import com.amqp.basic.queue.CommonConfigs;
4
5 //Run me - Third
6 public class CreateBindings {
7     public static void main(String[] args) throws IOException, TimeoutException {
8         ConnectionFactory factory = new ConnectionFactory();
9         factory.setHost("localhost");
10        factory.setUsername("ron");
11        factory.setPassword("ron");
12        factory.setPort(5672);
13        factory.setVirtualHost("/");
14
15        Connection connection = factory.newConnection();
16        Channel channel = connection.createChannel();
17
18        // Bind queues to the exchange
19        //Create bindings - (queue, exchange, routingKey)
20        channel.queueBind("topic1", "EduCostStatExchange", "Cost-[Year]-[State]-[Type]-[Length]");
21        channel.queueBind("topic2", "EduCostStatExchange", "Top5-Expensive-[Year]-[Type]-[Length]");
22        channel.queueBind("topic3", "EduCostStatExchange", "Top5-Economic-[Year]-[Type]-[Length]");
23        channel.queueBind("topic4", "EduCostStatExchange", "Top5-HighestGrow-[Years]");
24        channel.queueBind("topic5", "EduCostStatExchange", "AverageExpense-[Year]-[Type]-[Length]");
25
26        channel.close();
27        connection.close();
28    }
29 }
```

Based on the given requirements in assignment, I have used the routing key as the topic (from assignment question table column 3).

- Queues are >> topic 1,2,3,4,5
- Exchange is >> EduCostStatExchange

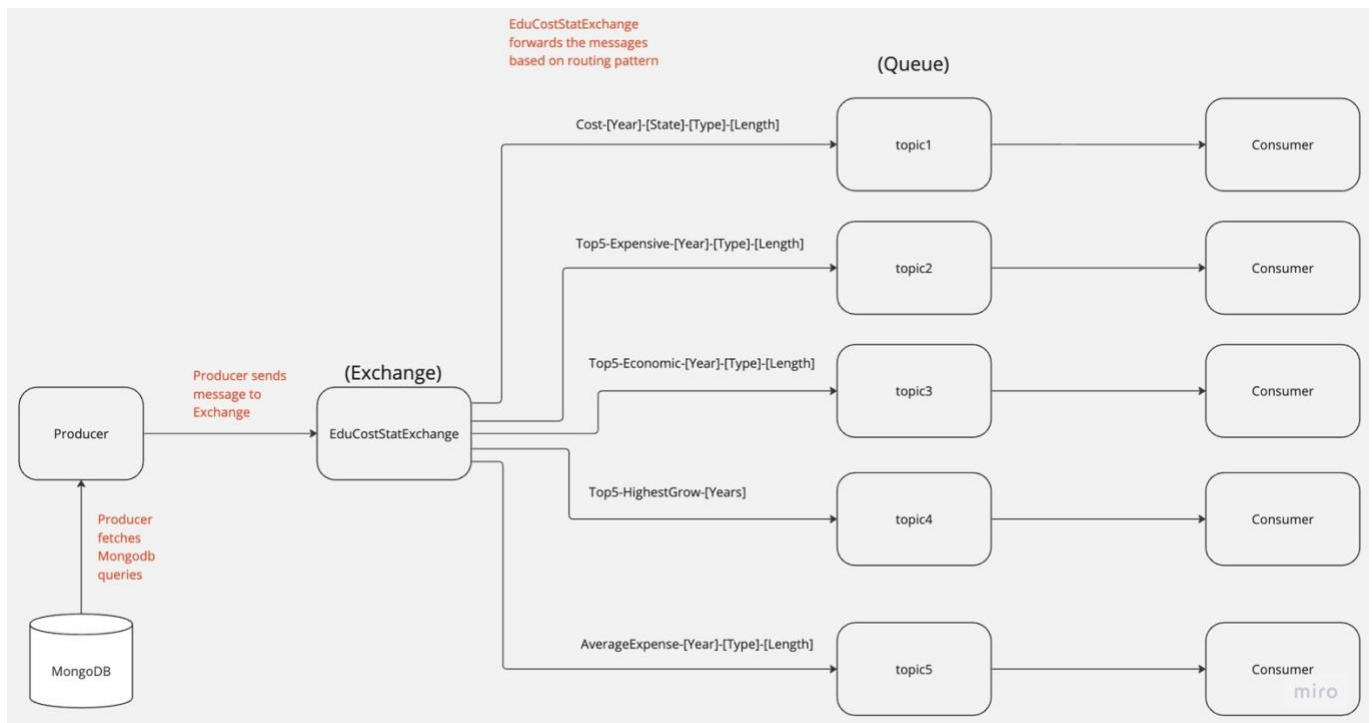
- Routing keys are >> Cost-[Year]-[State]-[Type]-[Length] , Top5-Expensive-[Year]-[Type]-[Length] , Top5-Economic-[Year]-[Type]-[Length] , Top5-HighestGrow-[Years], AverageExpense-[Year]-[Type]-[Length]

The bindings can be verified on RabbitMQ server as shown below –

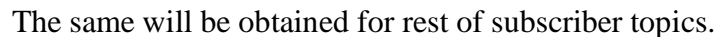
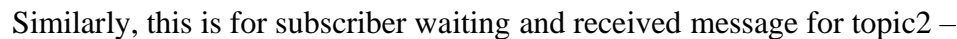
The screenshot shows the RabbitMQ Admin interface. The 'Bindings' tab is selected, showing a list of bindings for a specific exchange. The table below represents the data shown in the screenshot.

To	Routing key	Arguments
topic1	Cost-2013-Alabama-Private-4-year	Unbind
topic1	Cost-[Year]-[State]-[Type]-[Length]	Unbind
topic2	Top5-Expensive-2013-Private-4-year	Unbind
topic2	Top5-Expensive-[Year]-[Type]-[Length]	Unbind
topic3	Top5-Economic-2013-Private-4-year	Unbind
topic3	Top5-Economic-[Year]-[Type]-[Length]	Unbind
topic4	Top5-HighestGrow-2021	Unbind
topic4	Top5-HighestGrow-[Years]	Unbind
topic5	AverageExpense-2019-Public In-State-4-year	Unbind
topic5	AverageExpense-[Year]-[Type]-[Length]	Unbind

Finally, based on above codes and their workflow, below is the architecture for this code implementation –



Below is the subscriber console waiting and received message for the topic1 -



```

49 producer
50 Quick
51 rabbit-mq-java
52 rabbitmq-ping-project-master
53
54 -D getting-started
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
106
```

```
{
  "id": "6415543eb9fccaad4615563",
  "totalExpense": 48440,
  "State": "District of Columbia"
},
{
  "id": "6415543eb9fccaad4615564",
  "totalExpense": 48262,
  "State": "Connecticut"
},
{
  "id": "6415543eb9fccaad4615566",
  "totalExpense": 46114,
  "State": "Rhode Island"
},
{
  "id": "6415543eb9fccaad4615562",
  "totalExpense": 49871,
  "State": "Massachusetts"
},
{
  "id": "6415543eb9fccaad4615565",
  "totalExpense": 46255,
  "State": "Vermont"
}
```

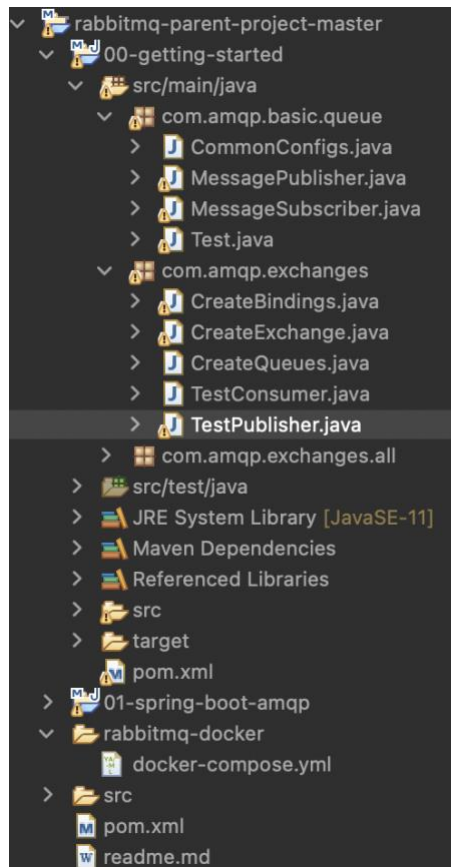
```
{
  "id": "Wyoming", "totalExpense": 13562}
{
  "id": "North Dakota", "totalExpense": 17742}
{
  "id": "Idaho", "totalExpense": 11544}
{
  "id": "Utah", "totalExpense": 15330}
{
  "id": "West Virginia", "totalExpense": 19120}
```

"id":	"Kentucky"	"expenses":	{["year": 2015, "expense": 21313, "rate": -0.2226774836456227}, {"year": 2013, "expense": 18586, "rate": -0.31592279855247285}, {"year": 2011, "expense": 17000, "rate": -0.25000000000000006}, {"year": 2009, "expense": 15000, "rate": -0.17647058823529413}, {"year": 2007, "expense": 13000, "rate": -0.15384615384615385}, {"year": 2005, "expense": 11000, "rate": -0.13636363636363635}, {"year": 2003, "expense": 9000, "rate": -0.11764705882352941}, {"year": 2001, "expense": 7000, "rate": -0.1}, {"year": 1999, "expense": 5000, "rate": -0.07692307692307693}, {"year": 1997, "expense": 3000, "rate": -0.05263157894736842}, {"year": 1995, "expense": 1000, "rate": -0.02631578947368421}, {"year": 1993, "expense": 0, "rate": -0.00000000000000001}, {"year": 1991, "expense": -1000, "rate": -0.02631578947368421}, {"year": 1989, "expense": -3000, "rate": -0.07692307692307693}, {"year": 1987, "expense": -5000, "rate": -0.11764705882352941}, {"year": 1985, "expense": -7000, "rate": -0.13636363636363635}, {"year": 1983, "expense": -9000, "rate": -0.15384615384615385}, {"year": 1981, "expense": -11000, "rate": -0.17647058823529413}, {"year": 1979, "expense": -13000, "rate": -0.2}, {"year": 1977, "expense": -15000, "rate": -0.22352941176470588}, {"year": 1975, "expense": -17000, "rate": -0.24705882352941177}, {"year": 1973, "expense": -19000, "rate": -0.27058823529411764}, {"year": 1971, "expense": -21000, "rate": -0.29411764705882354}, {"year": 1969, "expense": -23000, "rate": -0.3176470588235294}, {"year": 1967, "expense": -25000, "rate": -0.3411764705882353}, {"year": 1965, "expense": -27000, "rate": -0.3647058823529412}, {"year": 1963, "expense": -29000, "rate": -0.38823529411764705}, {"year": 1961, "expense": -31000, "rate": -0.4117647058823529}, {"year": 1959, "expense": -33000, "rate": -0.43529411764705883}, {"year": 1957, "expense": -35000, "rate": -0.4588235294117647}, {"year": 1955, "expense": -37000, "rate": -0.48235294117647056}, {"year": 1953, "expense": -39000, "rate": -0.5058823529411765}, {"year": 1951, "expense": -41000, "rate": -0.5294117647058824}, {"year": 1949, "expense": -43000, "rate": -0.5529411764705883}, {"year": 1947, "expense": -45000, "rate": -0.5764705882352941}, {"year": 1945, "expense": -47000, "rate": -0.6}, {"year": 1943, "expense": -49000, "rate": -0.6235294117647059}, {"year": 1941, "expense": -51000, "rate": -0.64705882352941177}, {"year": 1939, "expense": -53000, "rate": -0.6705882352941177}, {"year": 1937, "expense": -55000, "rate": -0.69411764705882354}, {"year": 1935, "expense": -57000, "rate": -0.7176470588235294}, {"year": 1933, "expense": -59000, "rate": -0.7411764705882353}, {"year": 1931, "expense": -61000, "rate": -0.7647058823529412}, {"year": 1929, "expense": -63000, "rate": -0.7882352941176471}, {"year": 1927, "expense": -65000, "rate": -0.8117647058823529}, {"year": 1925, "expense": -67000, "rate": -0.83529411764705883}, {"year": 1923, "expense": -69000, "rate": -0.8588235294117647}, {"year": 1921, "expense": -71000, "rate": -0.88235294117647056}, {"year": 1919, "expense": -73000, "rate": -0.9058823529411765}, {"year": 1917, "expense": -75000, "rate": -0.9294117647058824}, {"year": 1915, "expense": -77000, "rate": -0.9529411764705883}, {"year": 1913, "expense": -79000, "rate": -0.9764705882352941}, {"year": 1911, "expense": -81000, "rate": -1.0}, {"year": 1909, "expense": -83000, "rate": -1.0235294117647059}, {"year": 1907, "expense": -85000, "rate": -1.0470588235294117}, {"year": 1905, "expense": -87000, "rate": -1.0705882352941177}, {"year": 1903, "expense": -89000, "rate": -1.0941176470588235}, {"year": 1901, "expense": -91000, "rate": -1.1176470588235294}, {"year": 1899, "expense": -93000, "rate": -1.1411764705882353}, {"year": 1897, "expense": -95000, "rate": -1.1647058823529412}, {"year": 1895, "expense": -97000, "rate": -1.1882352941176471}, {"year": 1893, "expense": -99000, "rate": -1.2117647058823529}, {"year": 1891, "expense": -101000, "rate": -1.2352941176470588}, {"year": 1889, "expense": -103000, "rate": -1.2588235294117647}, {"year": 1887, "expense": -105000, "rate": -1.2823529411764706}, {"year": 1885, "expense": -107000, "rate": -1.3058823529411765}, {"year": 1883, "expense": -109000, "rate": -1.3294117647058824}, {"year": 1881, "expense": -111000, "rate": -1.3529411764705883}, {"year": 1879, "expense": -113000, "rate": -1.3764705882352941}, {"year": 1877, "expense": -115000, "rate": -1.4}, {"year": 1875, "expense": -117000, "rate": -1.4235294117647059}, {"year": 1873, "expense": -119000, "rate": -1.4470588235294117}, {"year": 1871, "expense": -121000, "rate": -1.4705882352941177}, {"year": 1869, "expense": -123000, "rate": -1.4941176470588235}, {"year": 1867, "expense": -125000, "rate": -1.5176470588235294}, {"year": 1865, "expense": -127000, "rate": -1.5411764705882353}, {"year": 1863, "expense": -129000, "rate": -1.5647058823529412}, {"year": 1861, "expense": -131000, "rate": -1.5882352941176471}, {"year": 1859, "expense": -133000, "rate": -1.6117647058823529}, {"year": 1857, "expense": -135000, "rate": -1.6352941176470588}, {"year": 1855, "expense": -137000, "rate": -1.6588235294117647}, {"year": 1853, "expense": -139000, "rate": -1.6823529411764706}, {"year": 1851, "expense": -141000, "rate": -1.7058823529411765}, {"year": 1849, "expense": -143000, "rate": -1.7294117647058824}, {"year": 1847, "expense": -145000, "rate": -1.7529411764705883}, {"year": 1845, "expense": -147000, "rate": -1.7764705882352941}, {"year": 1843, "expense": -149000, "rate": -1.8}, {"year": 1841, "expense": -151000, "rate": -1.8235294117647059}, {"year": 1839, "expense": -153000, "rate": -1.8470588235294117}, {"year": 1837, "expense": -155000, "rate": -1.8705882352941177}, {"year": 1835, "expense": -157000, "rate": -1.8941176470588235}, {"year": 1833, "expense": -159000, "rate": -1.9176470588235294}, {"year": 1831, "expense": -161000, "rate": -1.9411764705882353}, {"year": 1829, "expense": -163000, "rate": -1.9647058823529412}, {"year": 1827, "expense": -165000, "rate": -1.9882352941176471}, {"year": 1825, "expense": -167000, "rate": -2.0117647058823529}, {"year": 1823, "expense": -169000, "rate": -2.0352941176470588}, {"year": 1821, "expense": -171000, "rate": -2.058823
-------	------------	-------------	---

id	\$oid	"64193f7bf9ccad4943f3"	avgExpense	11451.555555555555	"Region"	"Northeast"	"Type"	"Public In-State"	"Length"	"4-year"
id	\$oid	"64193f7bf9ccad494332"	avgExpense	11120.133333333333	"Region"	"Southwest"	"Type"	"Public In-State"	"Length"	"4-year"
id	\$oid	"64193f7bf9ccad494334"	avgExpense	10489.333333333334	"Region"	"Mid-west"	"Type"	"Public In-State"	"Length"	"4-year"
id	\$oid	"64193f7bf9ccad494335"	avgExpense	10425.714285714286	"Region"	"East"	"Type"	"Public In-State"	"Length"	"4-year"
id	\$oid	"64193f7bf9ccad494334"	avgExpense	13346.5	"Region"	"Southwest"	"Type"	"Public In-State"	"Length"	"4-year"

For topic 5

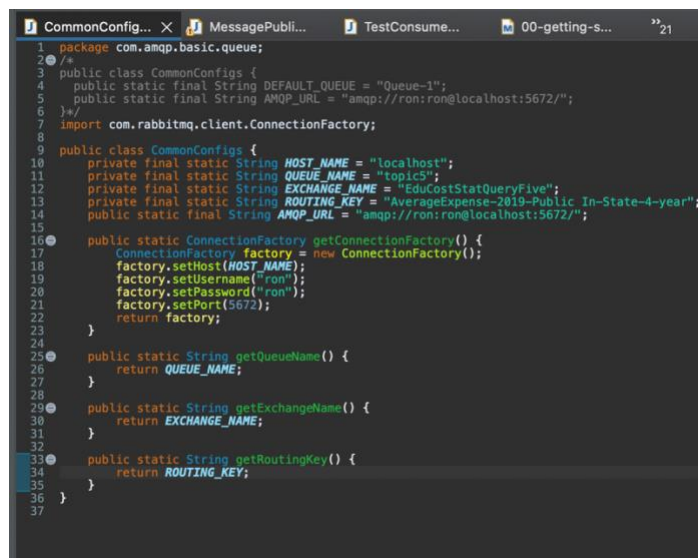
Codes location –



From above, we can conclude that -

The producer retrieves the datasets from each collection from the MongoDB cloud service for each topics listed in the table above. (this is done via MessagePublisher.java)

The parameters to customize each topic is set in a configuration file. –



The producer publishes the data to the exchange topics with a routing key that matches to the topic for each queue. (done via `MessagePublisher.java` , `CreateBindings.java`, `CreateExchange.java`, and `CreateQueues.java`)

The consumer receives the data from the queue based on the topic a consumer subscribed. (done via `MessageSubscriber.java`).