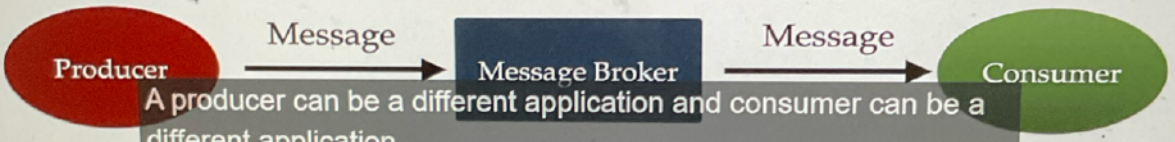
RabbitMQ

What is a Message Queue?

Message queuing allows applications to communicate by sending messages to each other. The message queue provides temporary message storage when the destination program is busy or not connected.

A message queue is made up of a producer, a broker (the message queue software), and a consumer.

A message queue provides an asynchronous communication between applications.



A producer can be a different application and consumer can be a different application.

RabbitMQ

RabbitMQ is a message queue software (message broker/queue manager) that acts as an intermediary platform where different applications can send and receive messages.

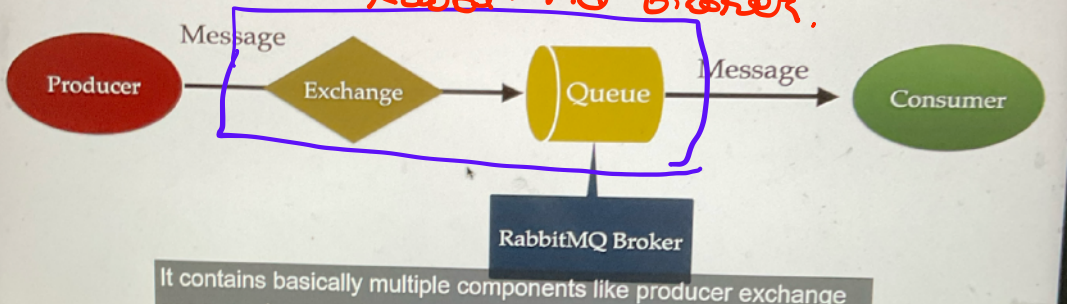
RabbitMQ originally implements the Advanced Message Queuing Protocol (AMQP). But now RabbitMQ also supports several other API protocols such as STOMP, MQTT and HTTP.

Well, rabbit is nothing but a message queue that acts as an intermediary platform where different applications

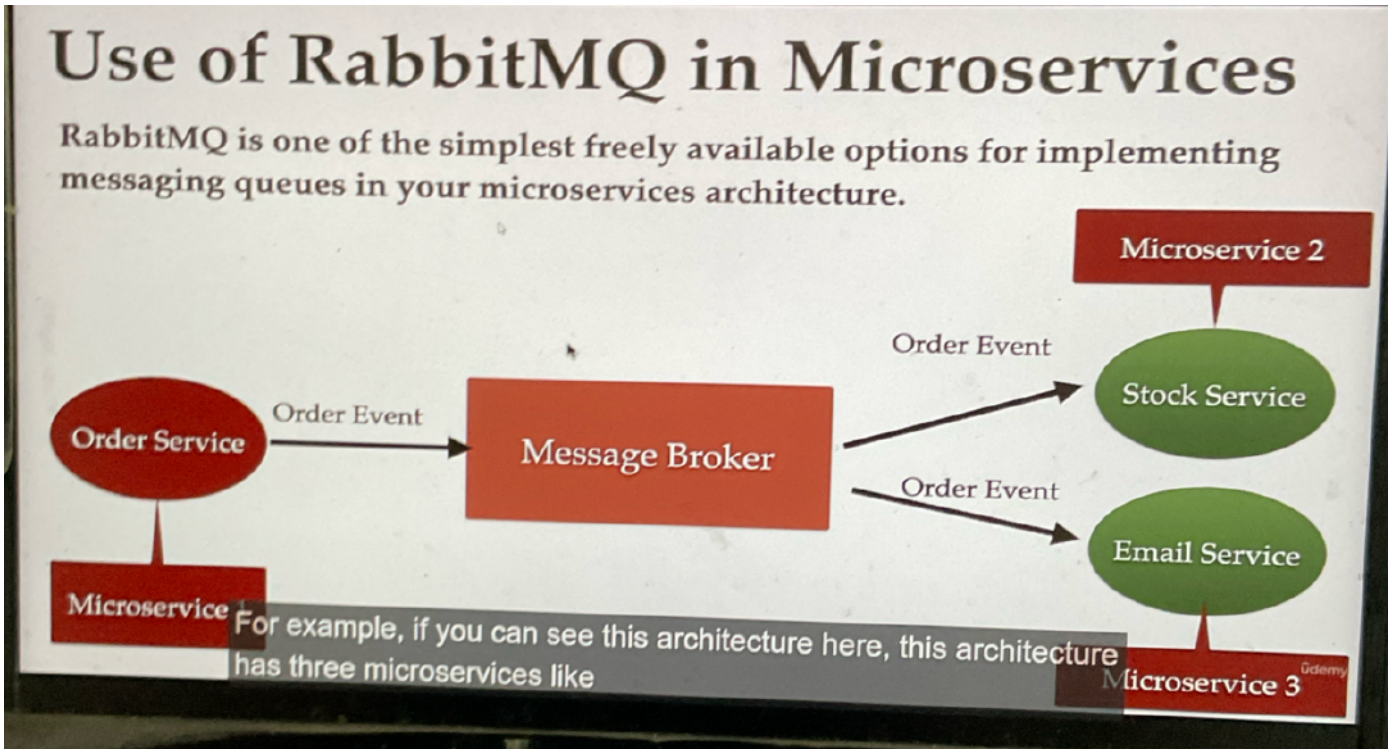
RabbitMQ

Producer is an application that sends messages to the RabbitMQ broker and Consumer is an application that reads messages from the RabbitMQ broker.

Rabbit MQ broker.



It contains basically multiple components like producer exchange queue and consumer.



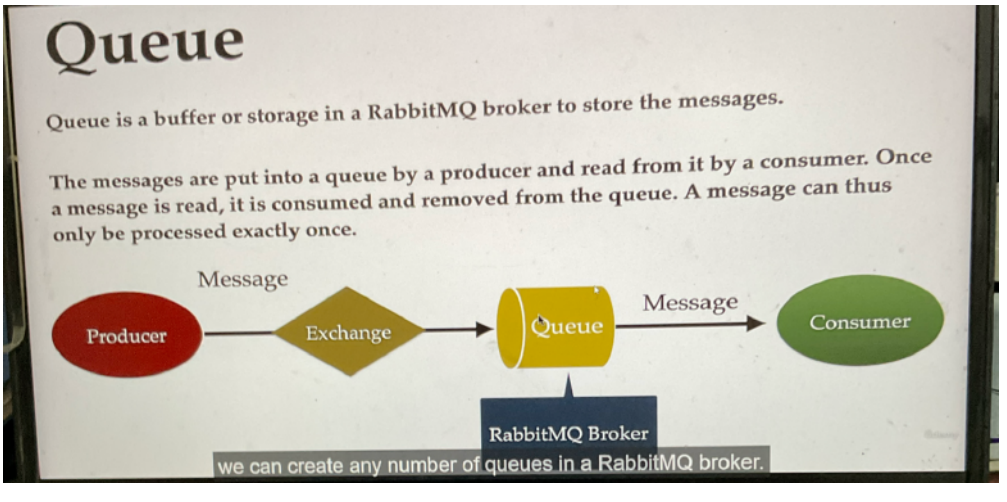
Producer : Producer sends message to broker.

↓

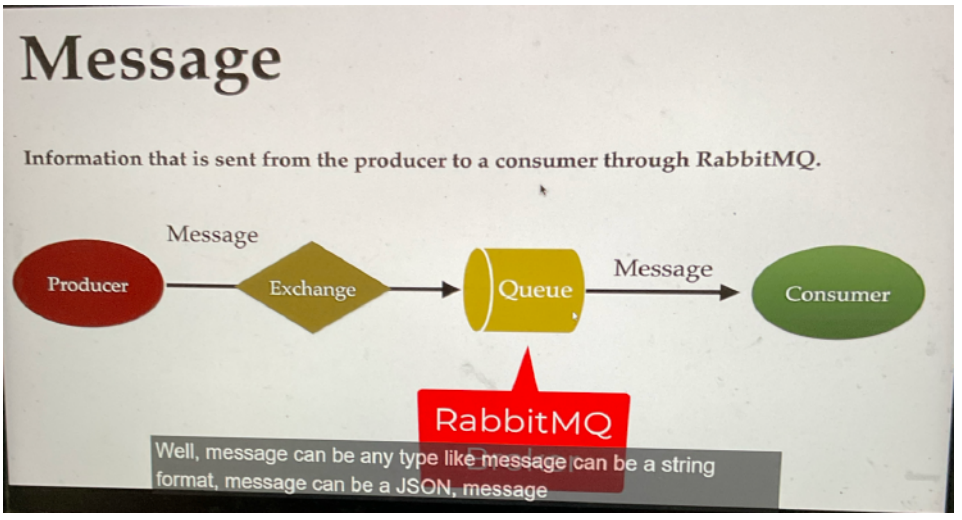
[Exchange + Queue]

Consumer : Consumer reads message from the RabbitMQ broker.

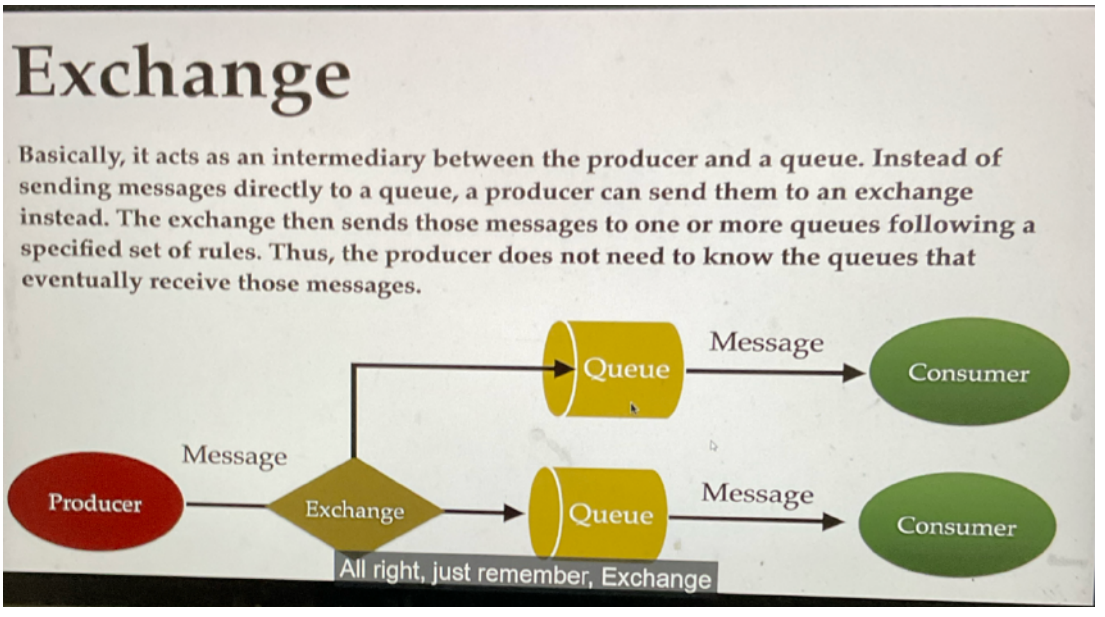
Queue :



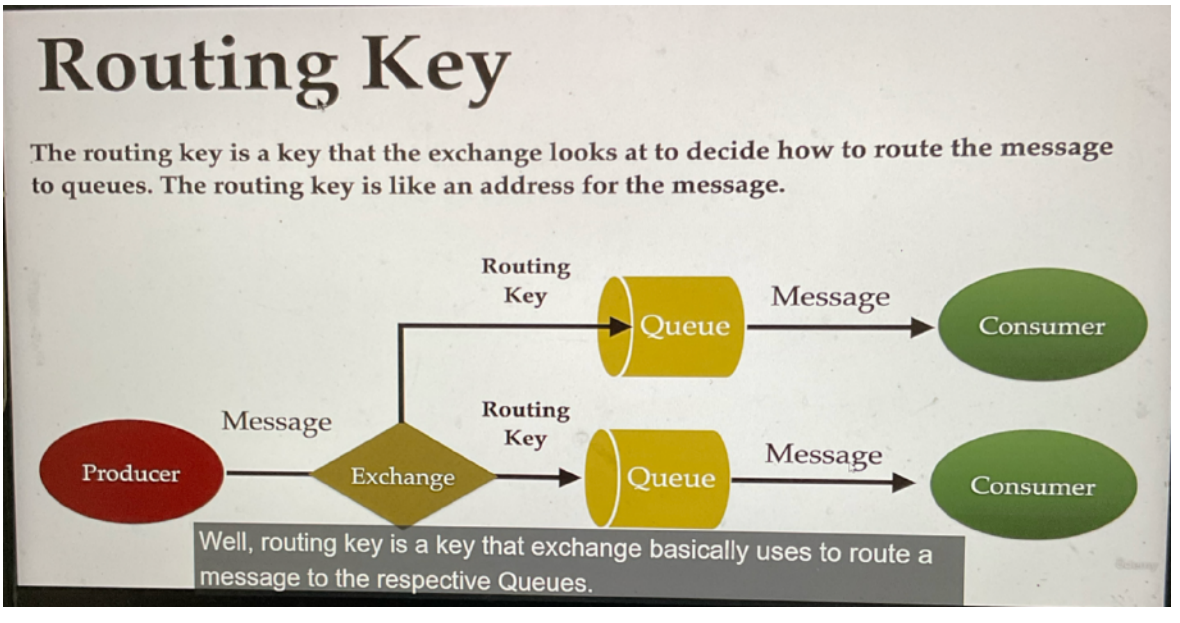
Message :



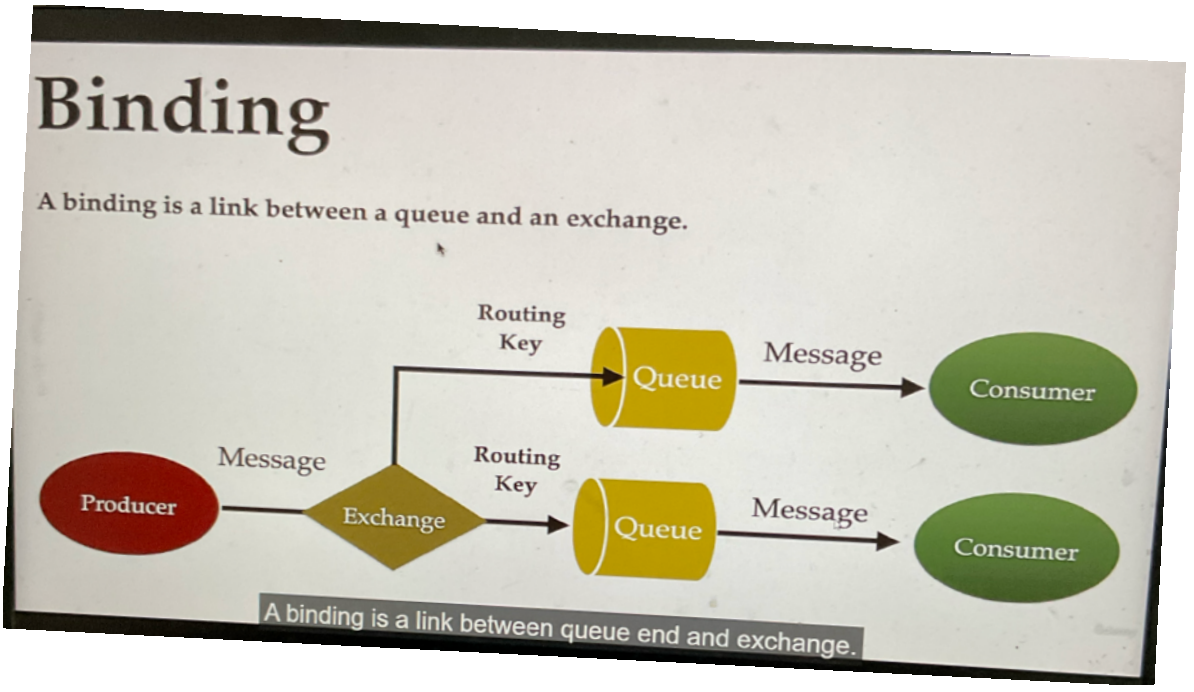
Exchange :



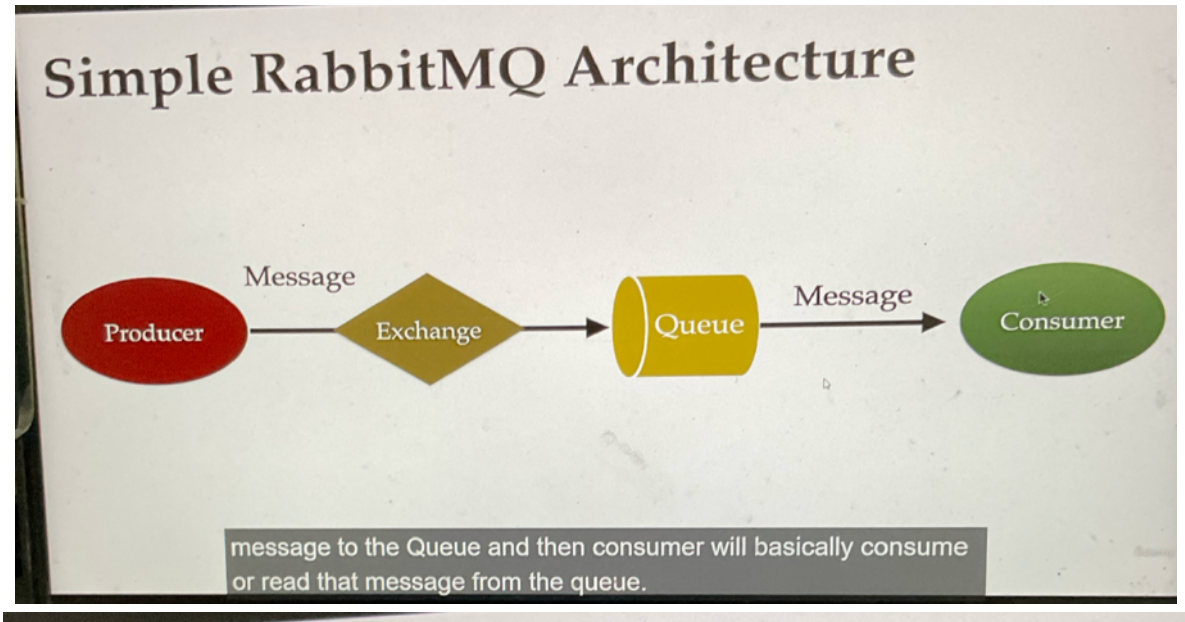
Routing Key :

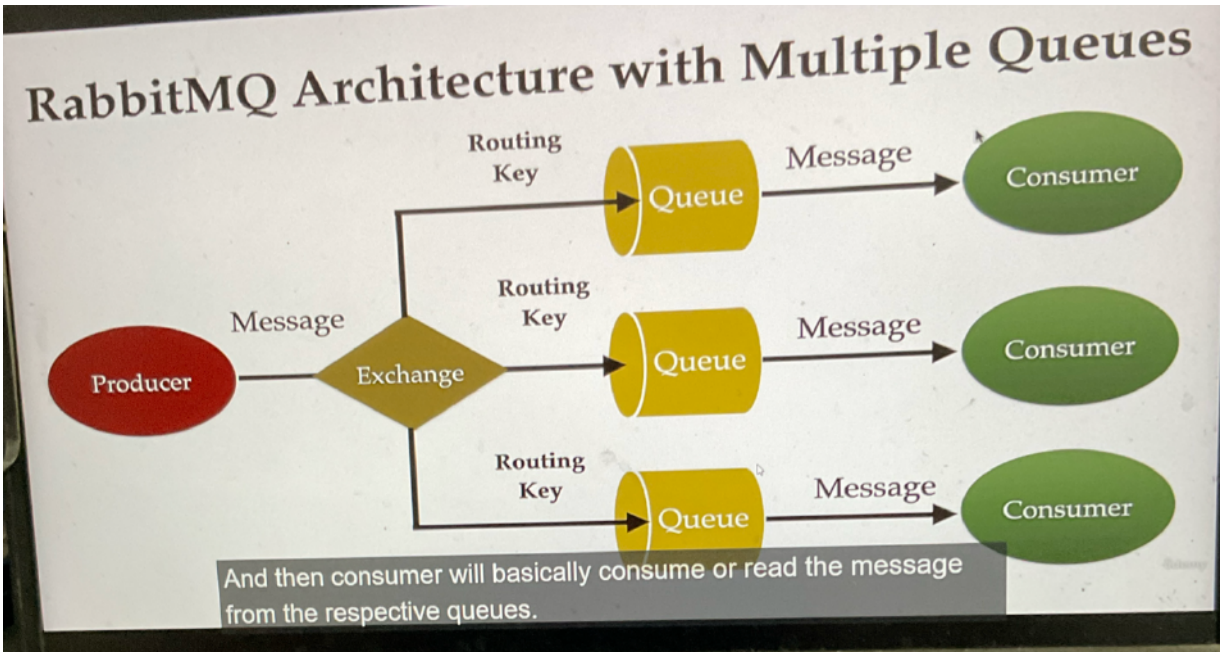


Binding :



RabbitMQ Architecture



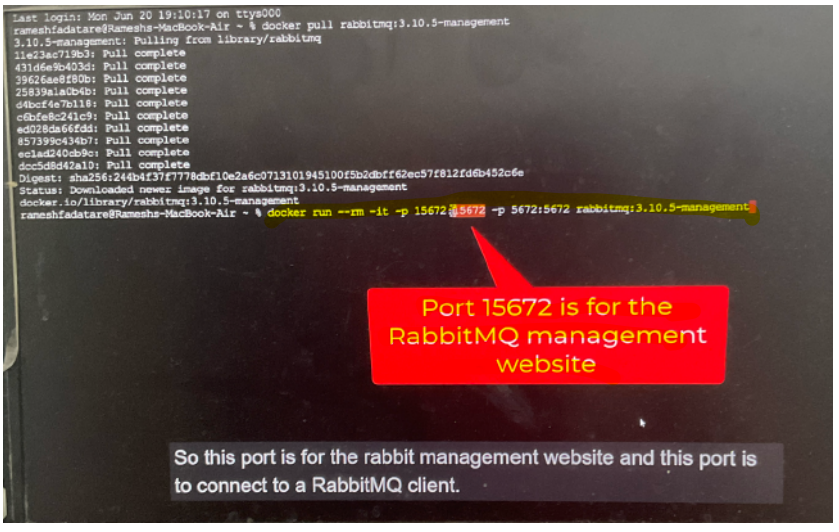


* Producer send a message along with Routing key. to the exchange.

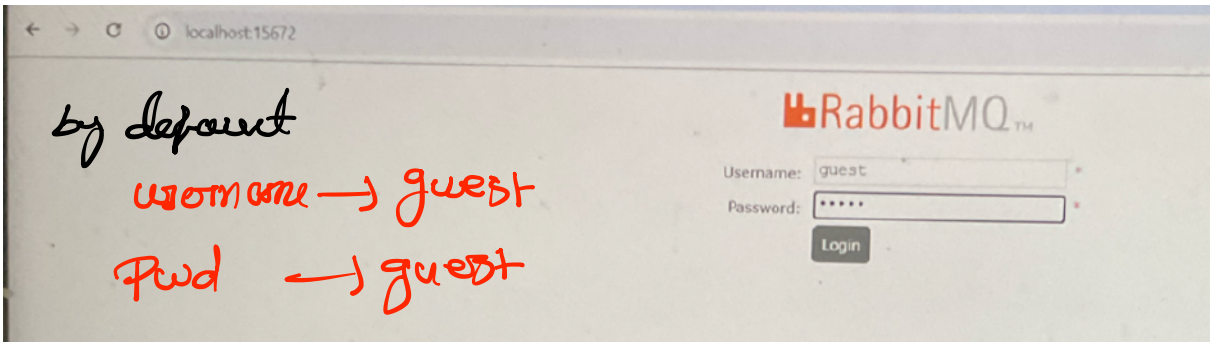
* Exchange use that Routing key send that message into respective Queue

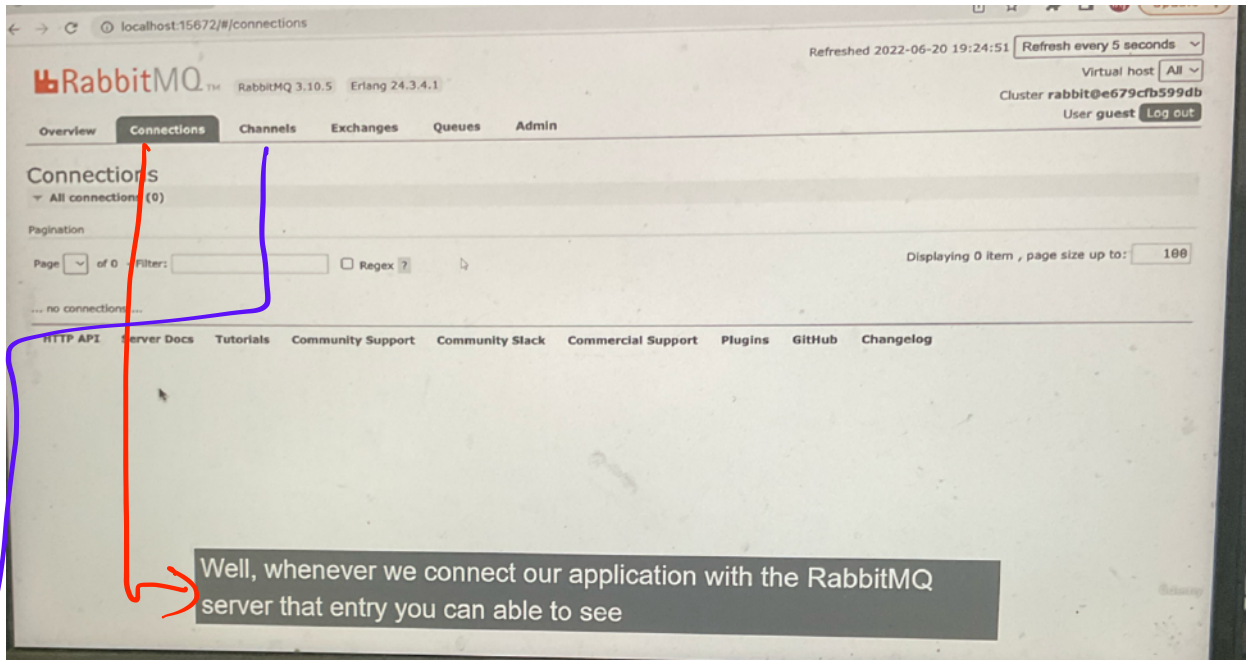
RabbitMQ :

Run :



RabbitMQ using Rabbitmq Management UI :

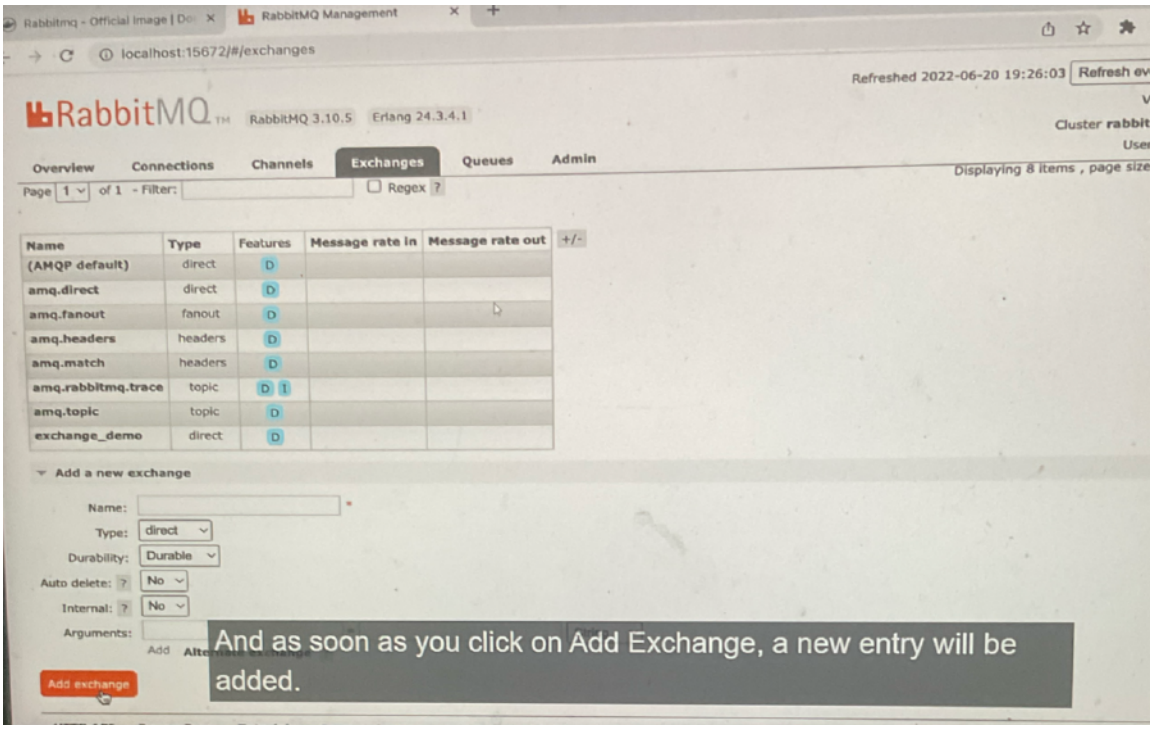




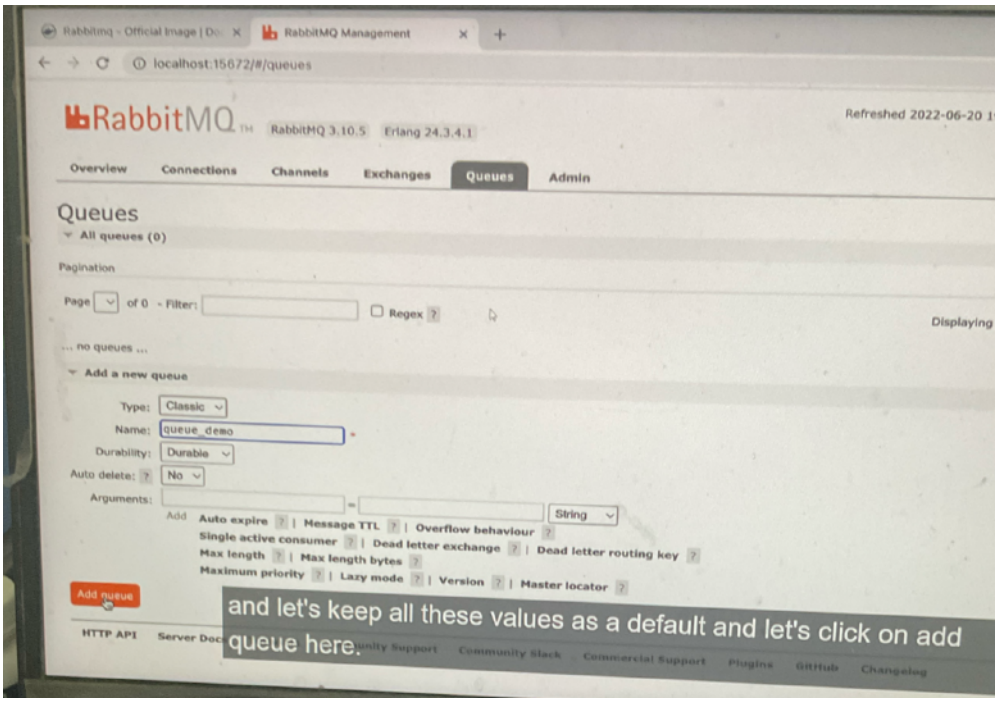
→ Producer and Consumer will basically exchange a message over a channel.

Exchange :

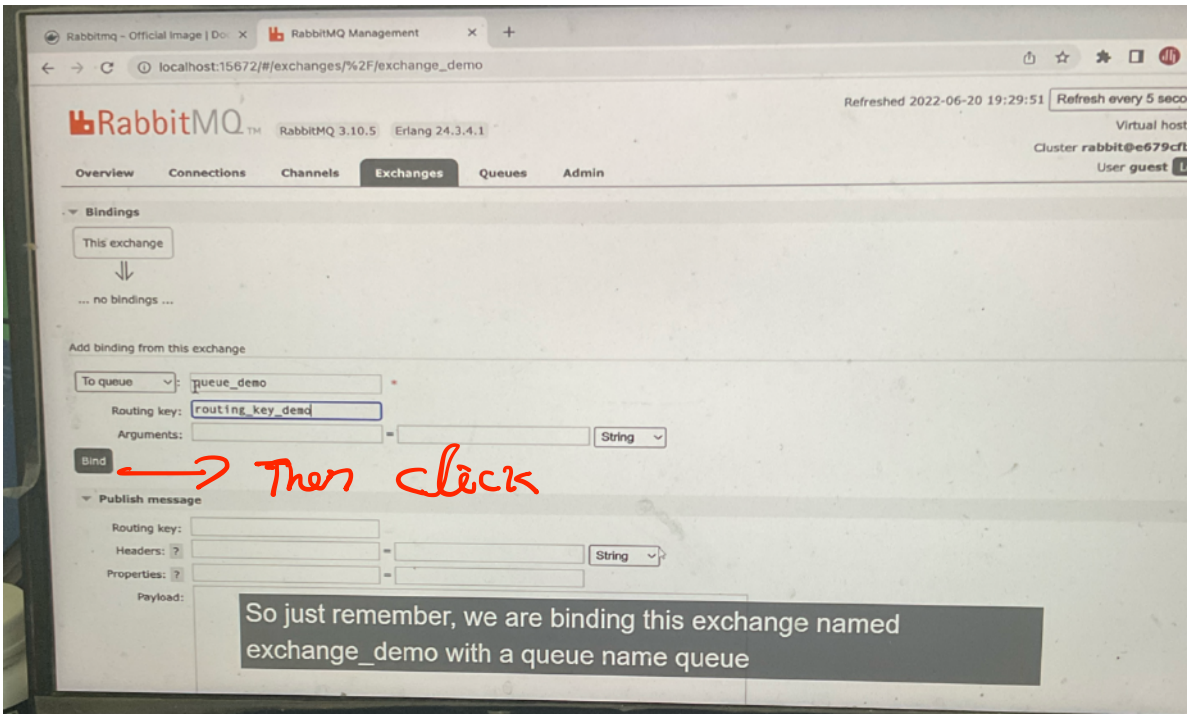
Create Exchange :



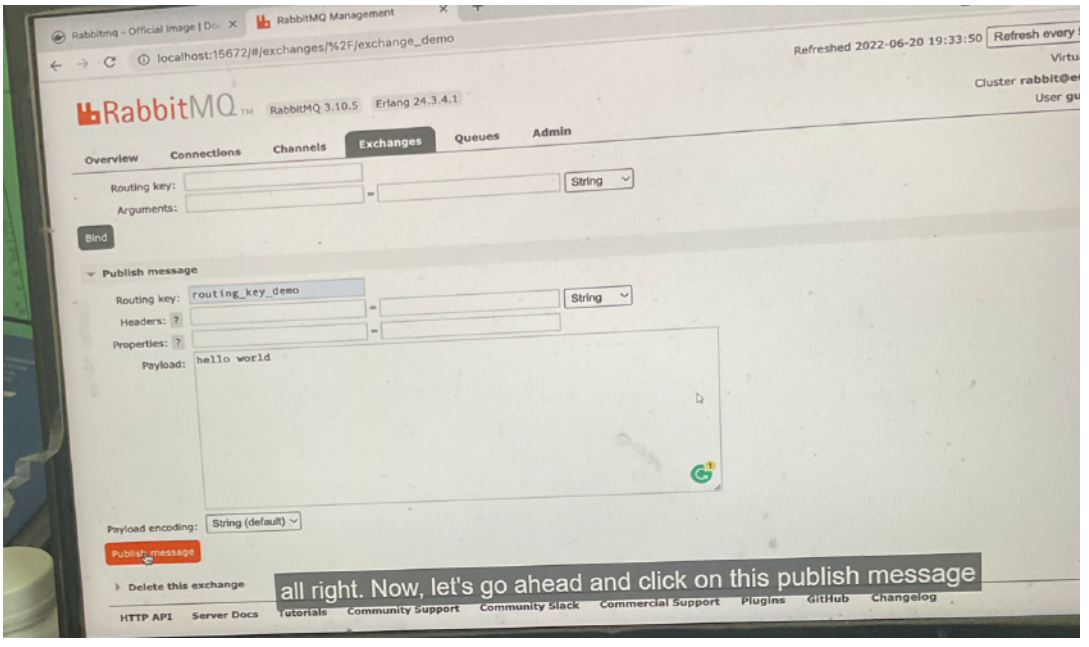
Creating Queue :



Binding queue and Routing key in Exchange :

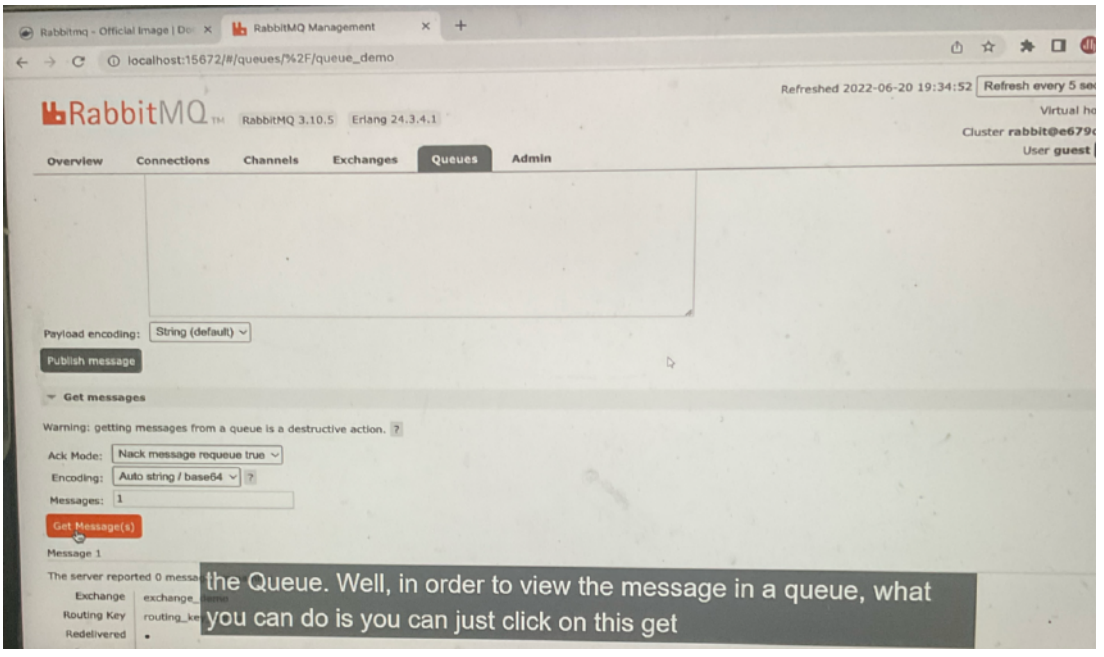


Sending Msg Exchange → Queue

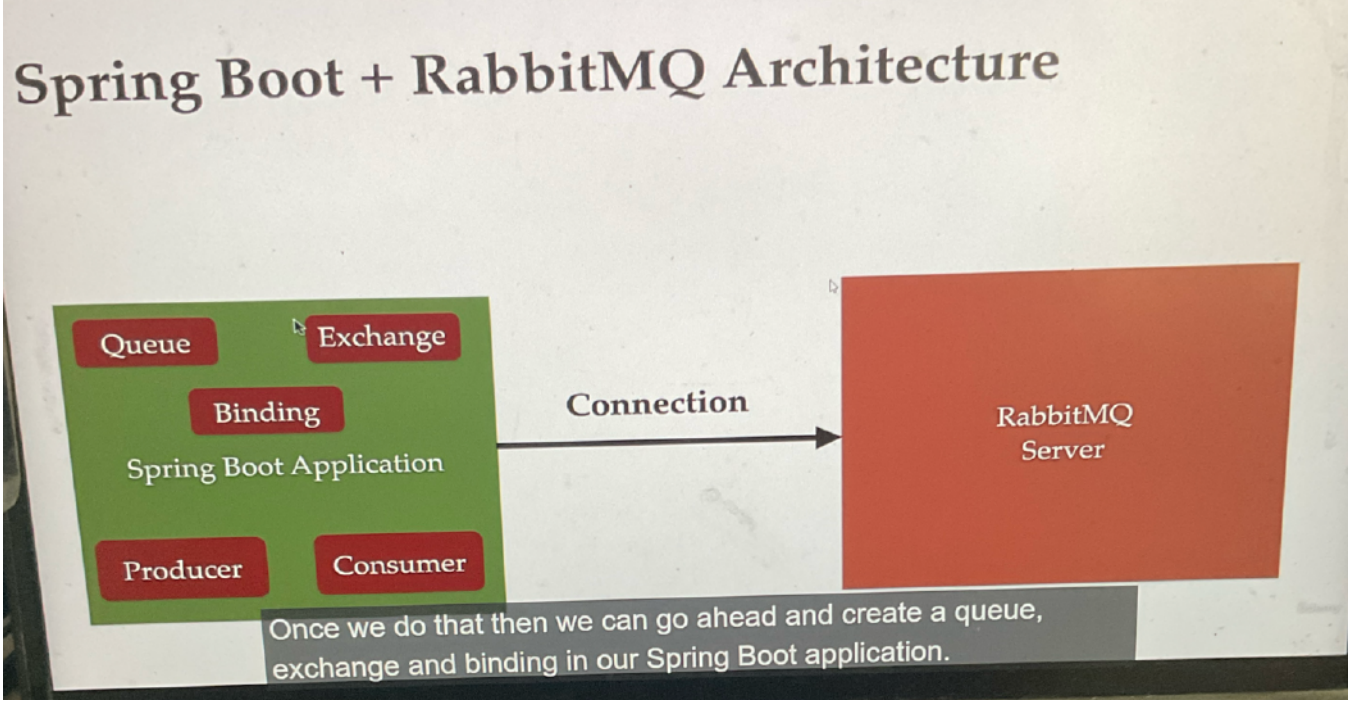


Read from Queue :

Click "get message to read from Queue".



Spring AMQP - advanced message Queuing Protocol



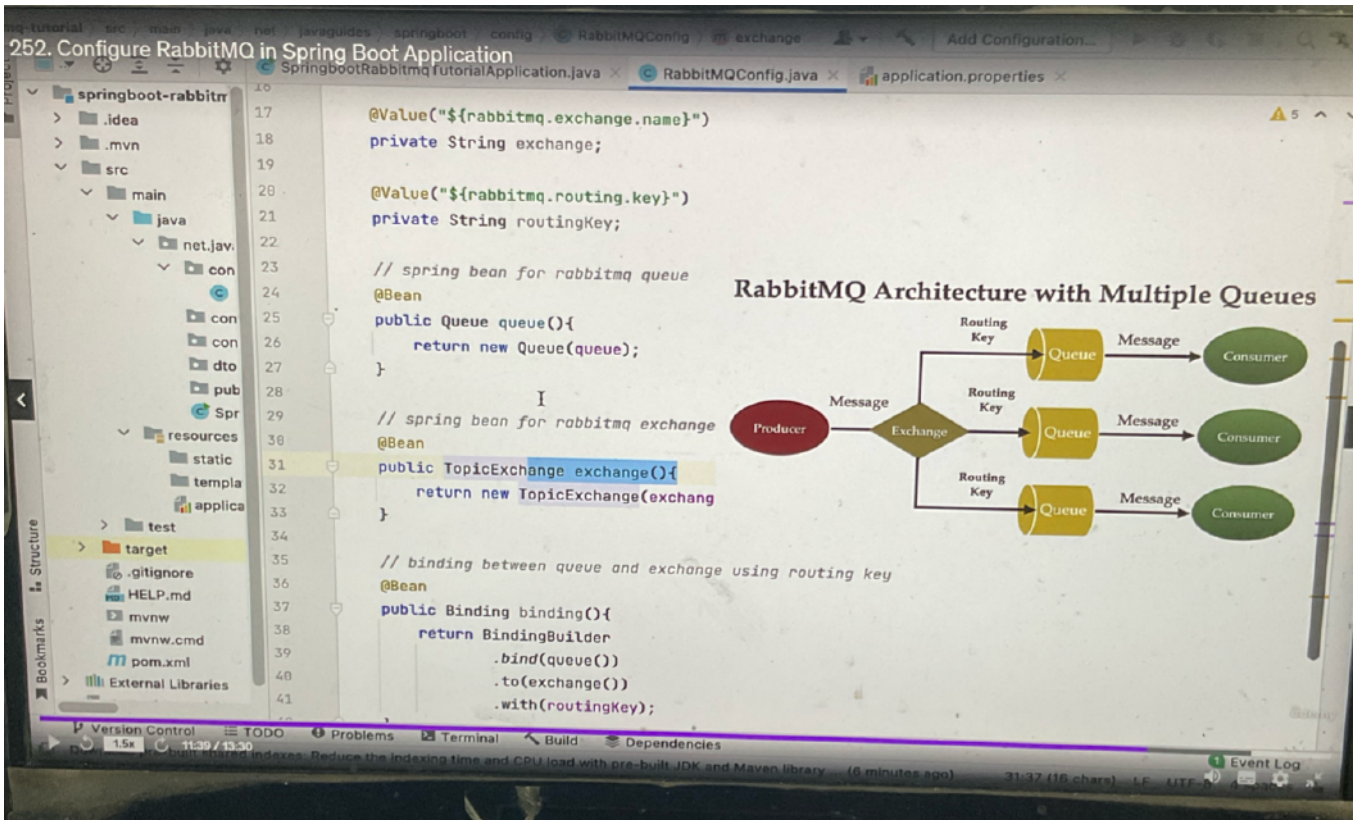
Spring Boot Autoconfiguration for Spring AMQP (RabbitMQ)

We get a connection to our RabbitMQ broker on port 5672 using the default username and password of "guest".

Define these properties in a application.properties:

```
spring.rabbitmq.host=localhost
spring.rabbitmq.port=5672
spring.rabbitmq.username=guest
spring.rabbitmq.password=guest
```

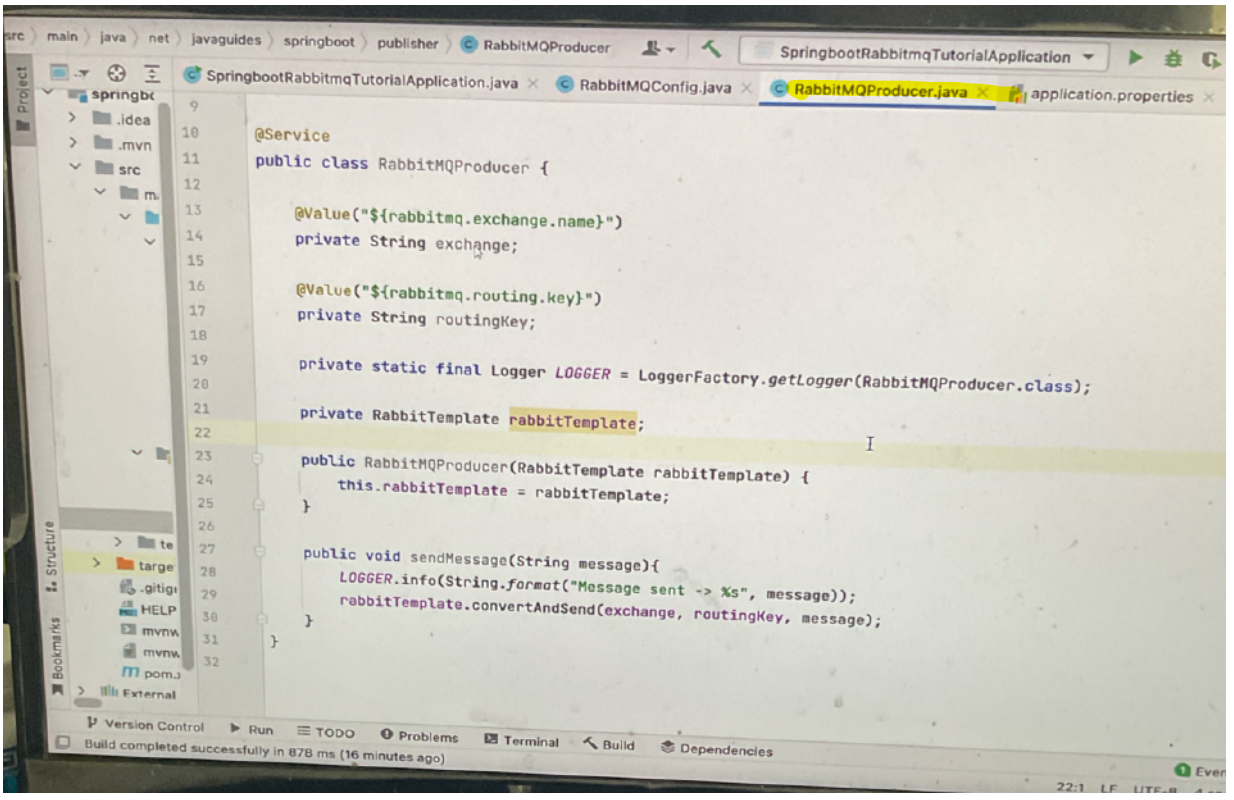
the RabbitMQ Broker on Port 5672 and uses the username password as a guest.



and few more Configuration needed

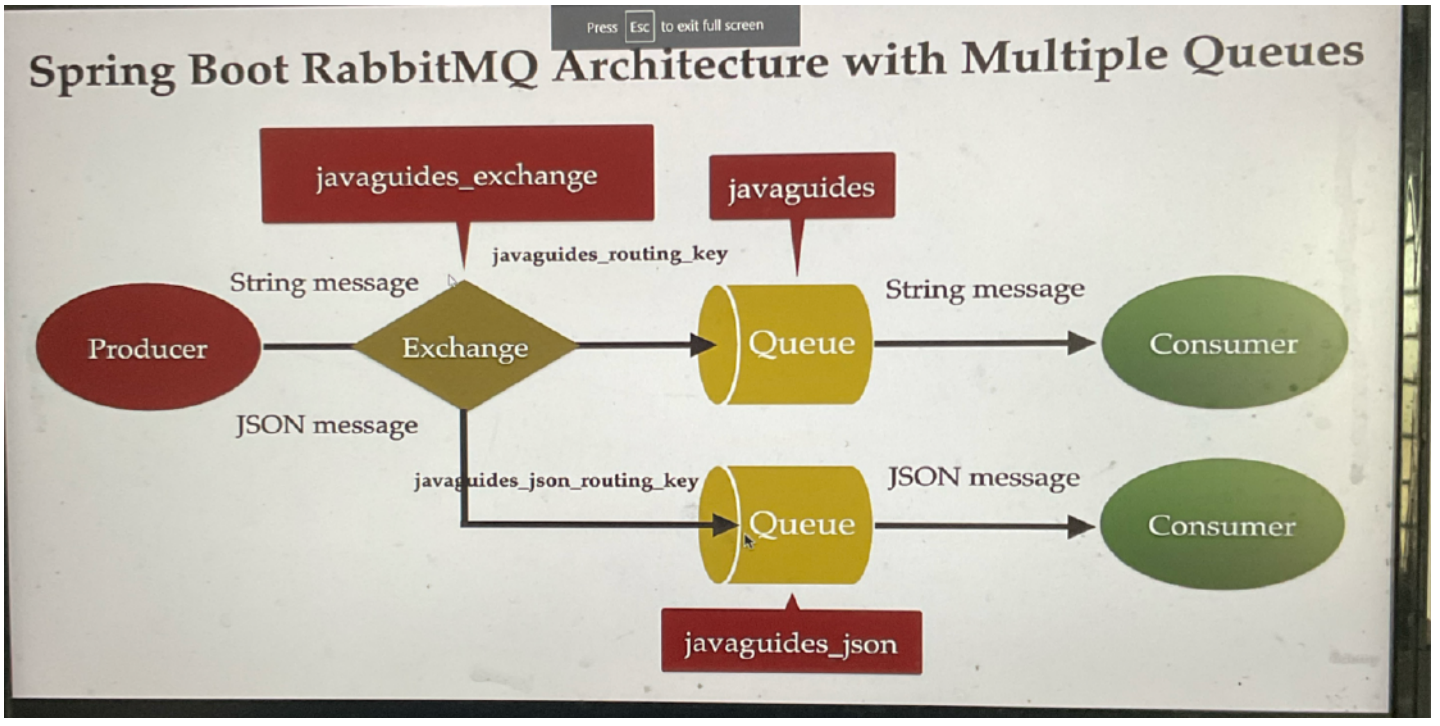
// ConnectionFactory
// RabbitTemplate
// RabbitAdmin

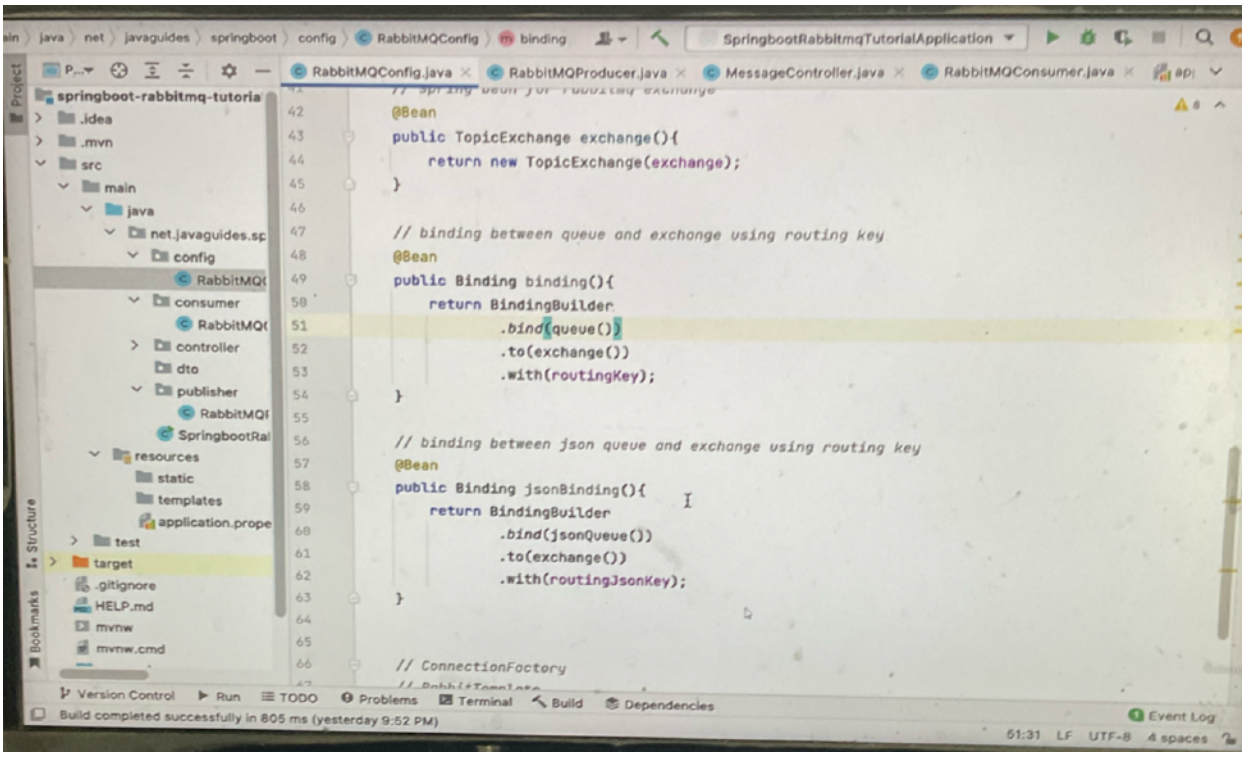
} RabbitMQ Auto Config take care of everything.

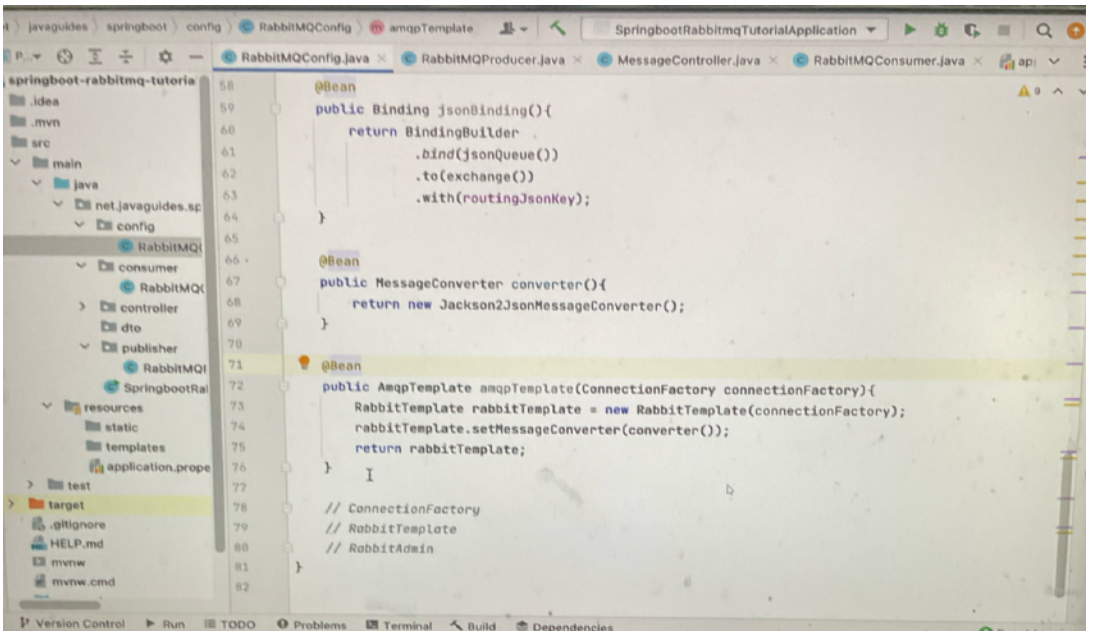


Sending Message:

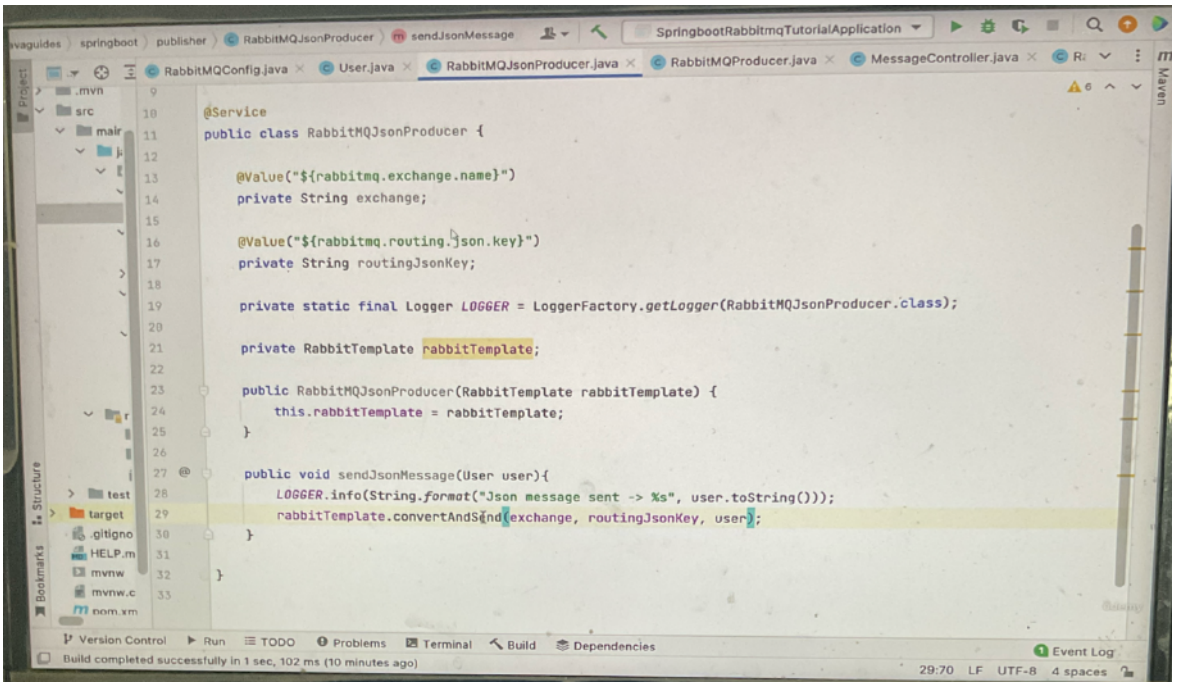
JSON Communication



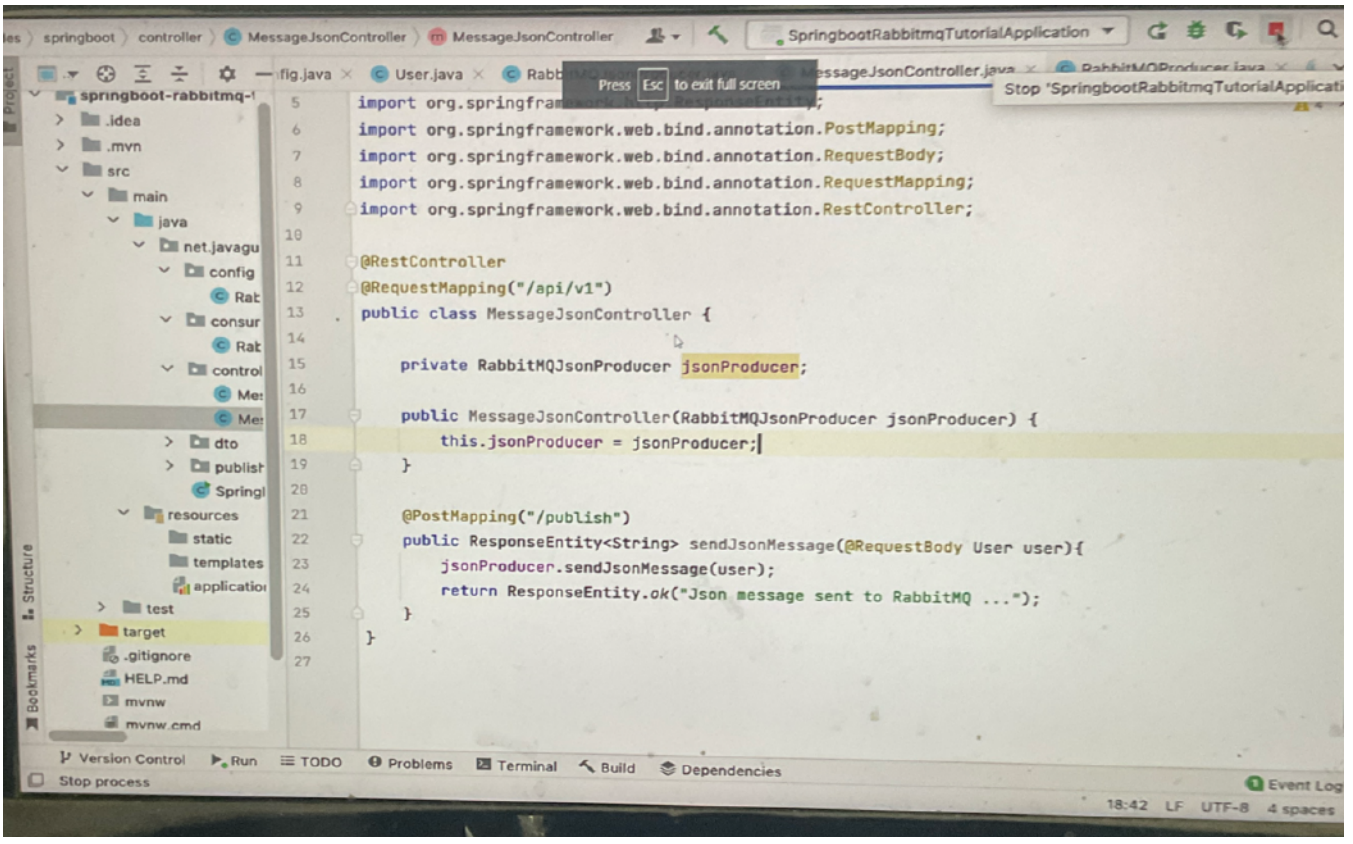



RabbitMQConfig.java
RabbitMQProducer.java
MessageController.java
RabbitMQConsumer.java
@Bean
public Binding jsonBinding(){
return BindingBuilder
.bind(jsonQueue())
.to(exchange())
.with(routingJsonKey);
}
@Bean
public MessageConverter converter(){
return new Jackson2JsonMessageConverter();
}
@Bean
public AmqpTemplate amqpTemplate(ConnectionFactory connectionFactory){
RabbitTemplate rabbitTemplate = new RabbitTemplate(connectionFactory);
rabbitTemplate.setMessageConverter(converter());
return rabbitTemplate;
}
// ConnectionFactory
// RabbitTemplate
// RabbitAdmin
}

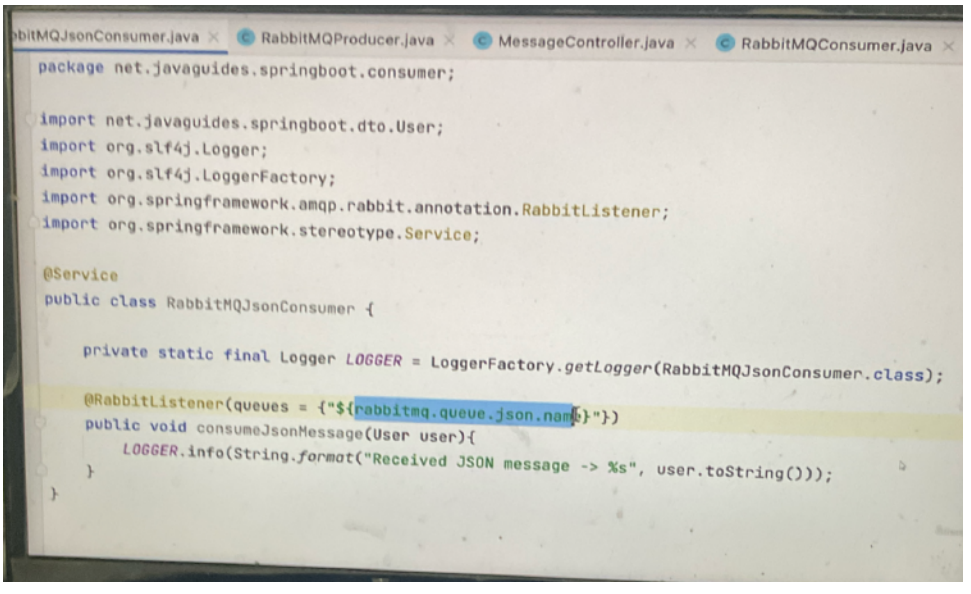
Producer :


RabbitMQConfig.java
User.java
RabbitMQJsonProducer.java
RabbitMQProducer.java
MessageController.java
@Service
public class RabbitMQJsonProducer {
@Value("${rabbitmq.exchange.name}")
private String exchange;
@Value("${rabbitmq.routing.json.key}")
private String routingJsonKey;
private static final Logger LOGGER = LoggerFactory.getLogger(RabbitMQJsonProducer.class);
private RabbitTemplate rabbitTemplate;
public RabbitMQJsonProducer(RabbitTemplate rabbitTemplate) {
this.rabbitTemplate = rabbitTemplate;
}
public void sendJsonMessage(User user){
LOGGER.info(String.format("Json message sent -> %s", user.toString()));
rabbitTemplate.convertAndSend(exchange, routingJsonKey, user);
}
}
Version Control
Run
TODO
Problems
Terminal
Build
Dependencies
Event Log
Build completed successfully in 1 sec, 102 ms (10 minutes ago)
29:70
LF
UTF-8
4 spaces

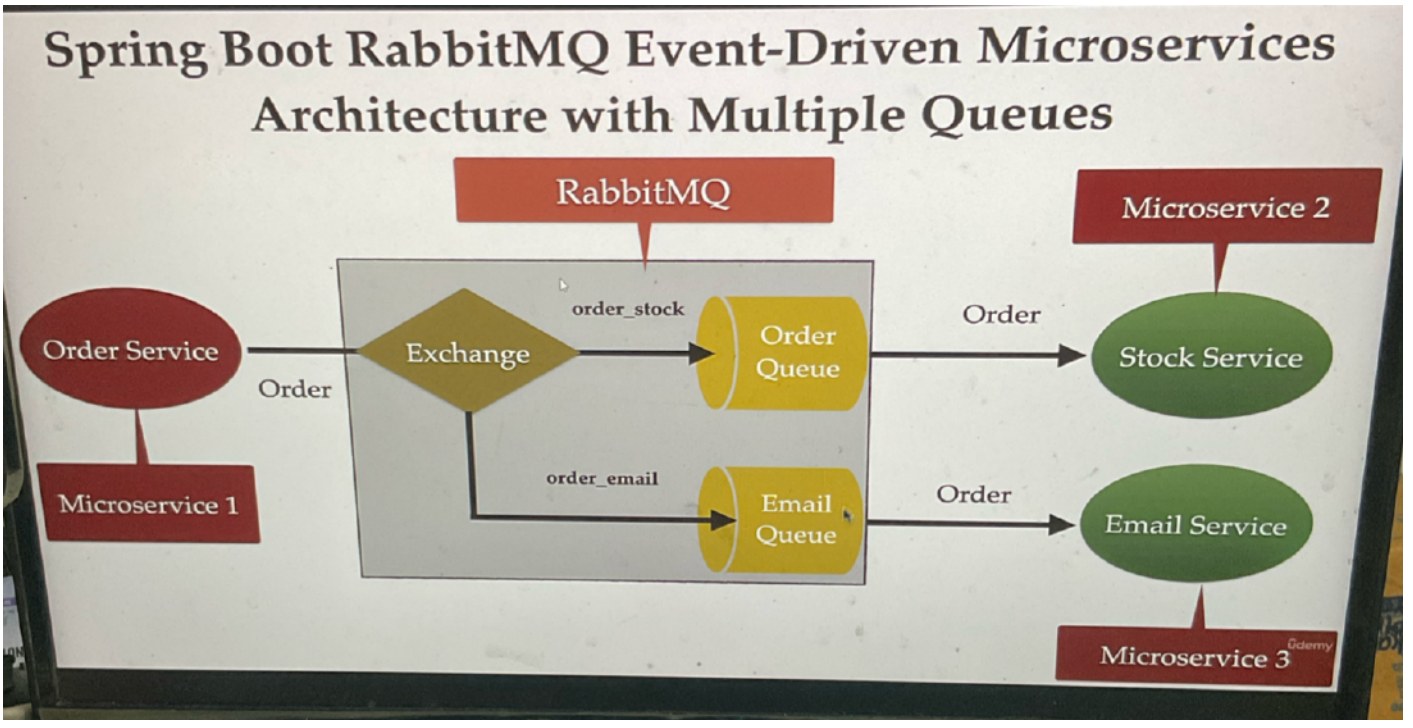
Controller Json :



Consumer Json :



Event Driven Architecture



Repo github : Dinesh3335

1) Order config

2) DTO

3) Producer

4) Controller

5) Consumer.