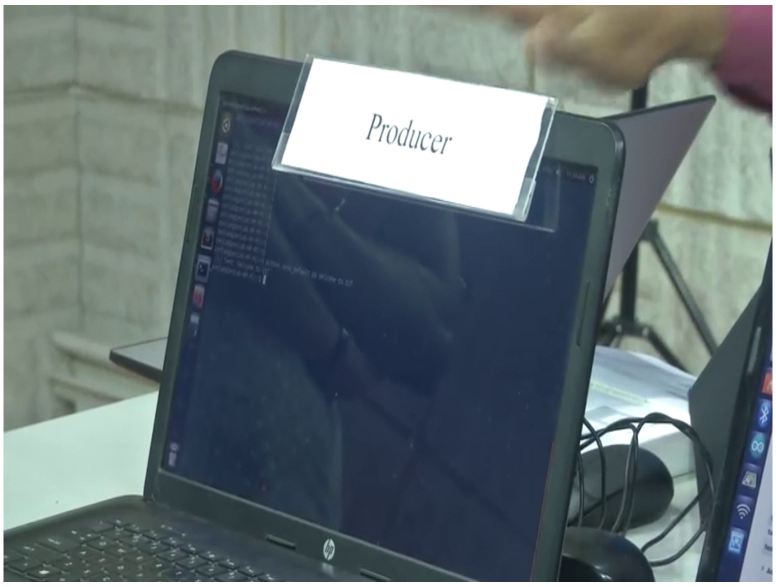
Design for Internet of Things

Prof. T V Prabhakar
Department of Electronic Systems Engineering
Indian Institute of Science, Bangalore

Lecture - 23
Introduction and Implementation of AMQP

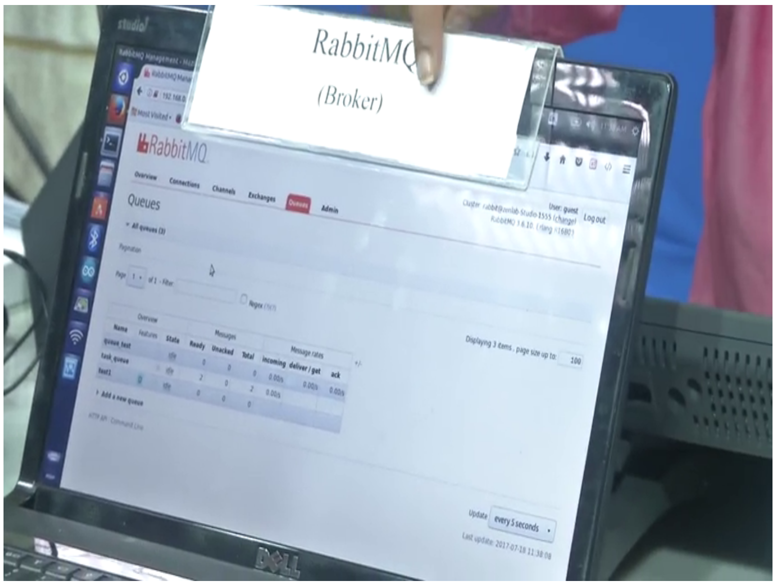
So, what we will do today is we will do some demonstrations related to another very exciting IoT protocol. And this broadly forms; the name of this protocol is AMQP-Advanced Messaging Queuing Protocol. This is a open standard again and it supports both point to point as well as publish subscribe models routing and queuing. Without any delay let us start with the demonstrations and then we look at all other related things with respect to the protocol.

(Refer Slide Time: 00:58)



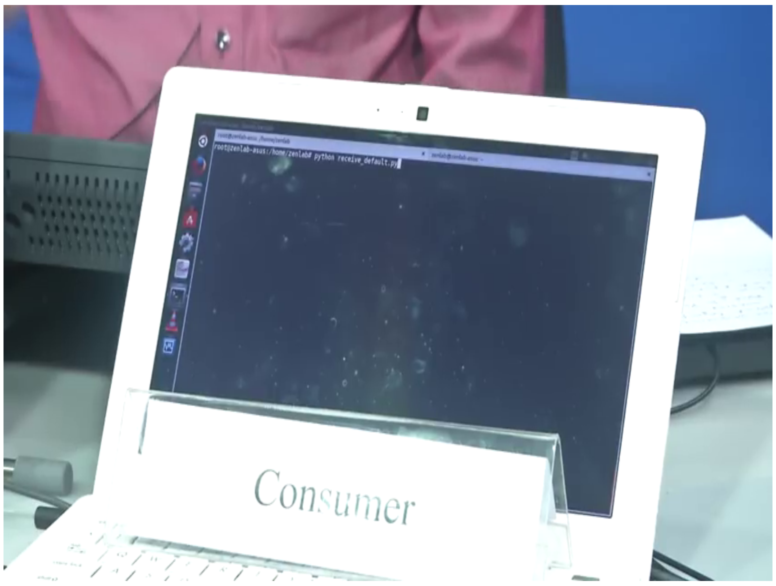
What I have on my the extreme right Is the publisher.

(Refer Slide Time: 01:03)



What I have here is what is known as a broker, look at the name it is called rabbit MQ.

(Refer Slide Time: 01:09)



```
root@zenlab-asus:/home/zenlab# python receive_default.py
```

And this is the subscribe. And unlike MQTT where basically there is a topic and there is a broker to which you subscribe or publish to the topic MQP is a little different compared to that. Basically all producers right to what are known as published what are known as exchanges and the exchange and depending on what is given in the routing Q that exchange will intelligently give it to different queues. So, the routing key is an additional thing. In fact, in MQTT the routing key itself in a way the topic right and if someone had the subscribe to it, it was just going there you will not be doing that anymore you publish to an exchange and the intelligence is from the exchange to the Q which is dependent on the routing key. So, to understand this better let us start the demonstrations.

The first demonstration that we have here is on the left side two of our colleagues Tejaswini and Abhirami will start the demonstration what we will do is you observe here carefully that there is a Q by name Q test and do not worry about what type of exchange this is this is one type of exchange where producer can directly write to the Q right. So, he is actually doing that. Now, she will type in the message which says python send default dot p y that is the script name welcome to IoT that is the what is the producer is trying to push to the broker there it goes. So, you can see here that this count should really go up by one you see that it has gone up by one why is it not 0 why is it one because the consumer was offline consumer did not want to pick it up, because for some reason he is not online. Now if he comes online the consumer should get this message right. So, let us do that and see what actually happens you can see that welcome to IoT what was published by the producer actually come here and the Q test has become 0. So, this is one type of exchange.

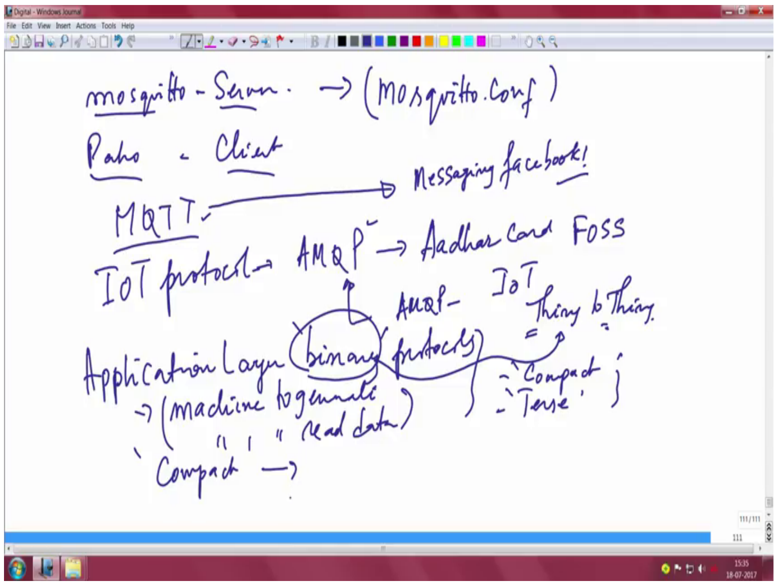
Essentially which will allow you to produce data and have consumers in this data now how did we set up this network this was set up by its connected wirelessly and basically you can replace these laptop that you have with handle devices like a mobile phones and basically have AMQP running also on very small devices these 3 are wirelessly connected there is an access point that has been placed here. And this access point is providing connections between the these producers rabbit I mean producers broker and the consumer the same model of producer not being in direct contact with the consumer continues to be an important thing quite similar to that of MQTT stop.

(Refer Slide Time: 04:31)

So, let us move on and see another demo subtle differences, but you have to note it very carefully what we did earlier with the producer was we run what is known as a script python script where we were invoking at direct type of exchange now what we are going to do is we are going to use fan out exchange fan out exchange simply means one to many. Now where are many here the consumer has we have invoke 2 consumers here just to show you that the one to many actually works and to see how that is done in a script let us try more just try opening that script Tejaswini, let us do just open; yes, let us see how the script looks yes you can see that the name of the exchange is logs and that appears here. You can see logs and you can also see that this is a fan out type of exchange which essentially means one to many and the type is mentioned here as fan out. So, these this is the key thing.

So, if she types a message here it should go to many and we should see the logs name of the exchange actually incrementing that. So, let us try that. So, we have now given python send fan out hello user one right. So, that is what has happened and you can see that it has appeared for user one and it has also appeared for user two. So, this is a mini demonstration of the capabilities of a rabbit MQ which essentially is implemented based on the advanced messaging queuing protocol.

(Refer Slide Time: 07:21)

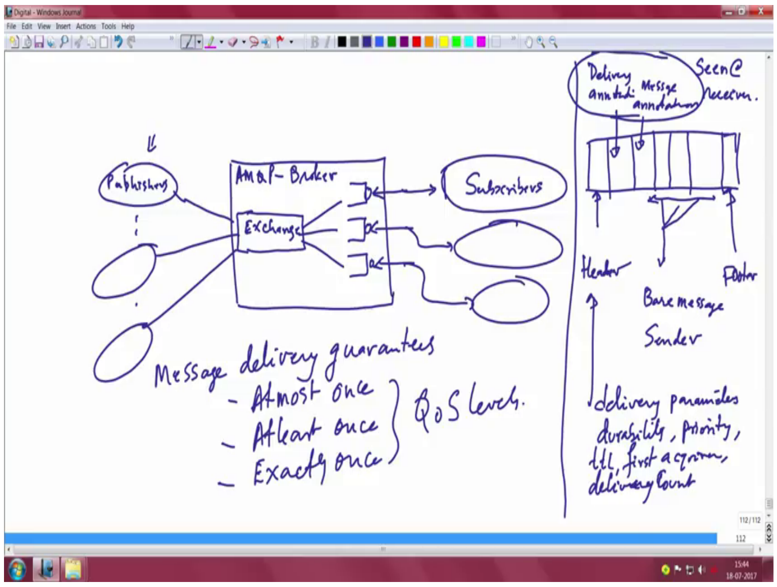

mosquitto - Server. → (mosquitto.conf)
Paho - Client
MQTT → Messaging facebook!
IoT protocol → AMQP → Aadhar card FOSS
AMQP - IoT
Thing to Thing
Application Layer (binary protocols)
Compact
Terse
→ (machine to generate
" " " read data)
Compact →

So, what I wrote on the screen is Paho client mosquito server and mosquito dot conf being the main file all this was with respect to all this was with respect to the MQTT right and now will have to move on and also covers sections related to another IoT protocol very very famous IoT protocol and very I would say robust one and this is called AMQP as I already mentioned to you during the recording of the video.

Now, you must note that if you had have if you have had a Facebook account you would have actually use MQTT because the messaging in Facebook actually uses MQTT and if you have had an Aadhar card. Aadhar card; the messaging protocol its one of the finest examples of what is free open source software right and AMQP is the messaging part in Aadhar card all of this means that these either if you take either MQTT or AMQP all these are application layer very important point application layer apply binary protocols why is this important because, these are meant for machines to read machines to generate machine to generate data and machine to read data you do not want humans to actually look up anything here things have to happen between machine to machine

Please recall that I mentioned to you that in the IoT world it is about thing to thing communication right thing to thing and not human to thing which is anywhere small part or human to human which is all very well known thing to thing essentially means that unless it is a binary protocol compact enters because it is not interpreted by humans it can be as much as possible. So, its compactors and that is how you get a lot of nice compression automatic compression because it is no I would not say compression I will say it is compact right its compact that compactness comes directly because of the fact that it is a binary protocol.

(Refer Slide Time: 10:40)



So, moving on how does the whole thing how does how does AMQP look and compare with that to the other well known pub sub let me draw a picture essentially a AMQP equalent broker AMQP equalent broker will actually have 2 parts one is the exchange. So, it is written ex like this and then there are what are known as queues you can imagine something like this I mention to you about the type of exchanges direct exchange fan out exchange and so on. So, these are the publishers and these are basically subscribers. So, this essentially forms the mechanism in AMQP right. So, you can say that this is a message oriented protocol and there are message delivery guarantees starting with at most once at most once at least once as well as exactly once the Q S levels which we know very well also applicable to this protocol. So, that is one thing.

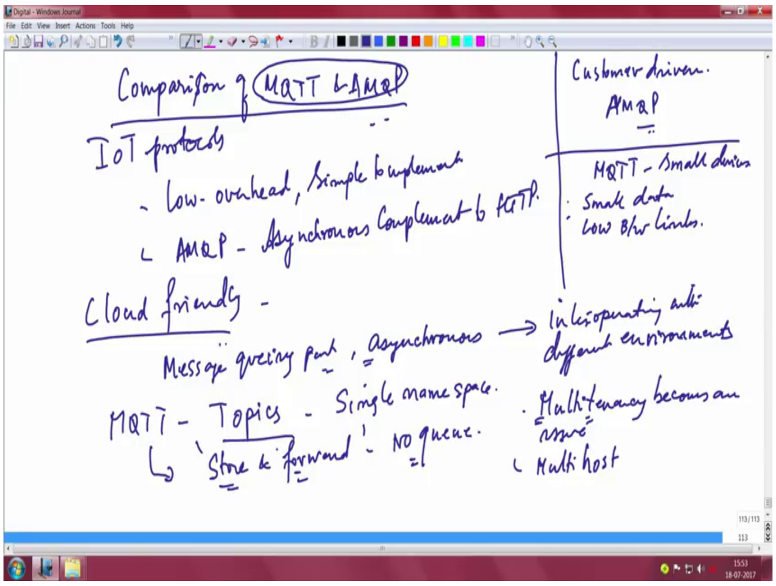
The message format appears a little strange and maybe I should quickly write down how it looks basically any message will have a header and a footer. So, let us put down the header and footer there are what are known as delivery annotations to this can be typically a delivery annotation and this is a message annotation then there are also what are known as fields for bare message. So, you can have you can have bare message. So, this is a bare message as well.

So, essentially what you see here in this picture is the fact that both bare messages as well as annotated message are possible within the message format of AMQP and it is. So, this messaging is on top of a transport layer the messaging capabilities are handle all possible message and capabilities are handled in this layer and AMQP. As I mentioned has this 2 types of messages these bare messages are basically those are supplied by the sender those which are supplied by the sender and annotated messages. Those which are actually seen at the receiver this is all seen at the receiver oops looks anyway this is important at the receiver seen at the receiver at receiver.

Now, the header in this format actually conveys the delivery parameter that is interesting right it conveys all the delivery parameters. So, what does it mean it means durability then priority time to live first acquirer first acquirer? And also finally, delivery count all of this is there in the header part the transport layer provides the required extension points for the message layer. And in this layer the communications are all basically frame oriented. And there is again a structure for the AMQP frames which I will not draw, but there is indeed a structure and this structure is what is well defined in the standard.

So, I think basically if you understand that AMQP is also has the ability of using publish subscribe in addition to other methods of delivery it is not only just about broker publish and subscribe, but its more is more than that in the sense that it is both point to point publish subscribe model routine queuing and so on right. So, that is the key thing here.

(Refer Slide Time: 18:40)



Comparison of MQTT & AMQP

IoT protocols
- Low-overhead, Simple to implement
- AMQP - Asynchronous Complement to HTTP.

Customer driven.
AMQP

MQTT - Small devices
: Small data
: Low B/w links.

Cloud friendly -
Message queuing part, asynchronous → interoperating with different environments

MQTT - Topics - Single name space.
↳ 'Store & forward' - No queue.

. Multitenancy becomes an issue
. Multi host

So, often we are asked this question what is the comparison? Comparison of MQTT and AMQP right this is what people keep asking and I think it is quite natural that you should understand major differences between them well both are IoT protocols which you should know. I have already said that MQTT is more like I would say low overhead is more like a low overhead simple very simple to implement very simple to implement both for sending data for receiving data and so on AMQP is you can say is more like as heavy as http.

And I would perhaps even called it; it is perhaps a asynchronous version compliment I will say of complement to http you can say more or less a like that see the good thing about both these is that they are very cloud friendly. Remember our initial introduction to IoT we did talked about storage and we did talk about cloud server and or in the cloud on the cloud systems if you wish to run MQTT or AMQP it is quite easy to install them on cloud system.

So, they are very cloud friendly. So, cloud computing for instance becomes much simpler the message queuing with its asynchronous nature basically the message queuing part queuing part and this fact that it is asynchronous right part is a perfect for inter operating many different environments. So, very perfect allows you for lot of inter operating inter operating with different environments.

So, that is really very good right. So, all of MQTT if you take I will split between MQTT and AMQP as we keep explaining everything around MQTT is about this topic and topic itself is like a single namespace right it is a single namespace and you do not have. For example, one major I would say short coming is that store and forward is not known to MQTT they are nothing like store and forward. And there are nothing like queues no queues actually if you have a queue then you can associated with sorry about this. So, because there are no queues store and forward is not there.

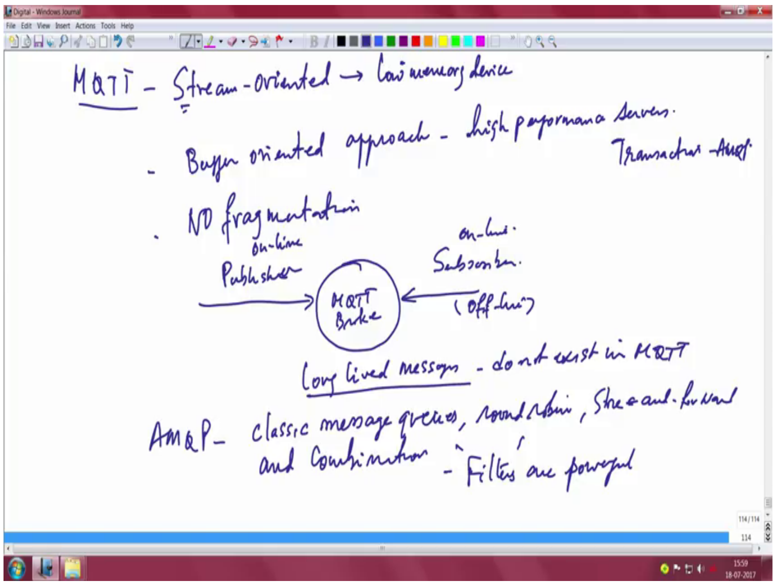
So, if this is not there what is a big deal multi tenant multi tenant or tenancy becomes an issue if you do not have these capabilities in other words a single piece of software serving a set of hosts the single instance not set up a single instance of AMQP can serve a set of hosts. And that is what will give it this multi tenancy feature which is not possible in MQTT you have a single broker all of them connect to that broker right.

So, that way AMQP is quite a multi tenant multi tenant approach is there and multi hosts is also possible multi hosted. So, you can just imagine how much it will scale when you we work with multi hosting multi host capability also is there. So, this is a big thing in AMQP further things are this is not really a company driven not company driven which one AMQP its more customer driven customer driven paradigm that is the key thing.

And if you look at any major big large projects and corporate, you will find that it is all about AMQP installation, but hardly you will find anything in the MQTT space. So, that is the another important thing which is quite obvious from the fact that the standard is open and also the fact that Aadhar card was actually using it right is actually using this as a messaging protocol. So, that is important.

If you turn back to MQTT again go back and look at what MQTT does its mostly for small I would say small devices its meant for small devices small sending small data small data over constraint links right over low bandwidth links over low bandwidth links, but AMQP full messaging scenario is possible. So, it is really exhausted in that sense.

(Refer Slide Time: 26:21)



MQTT – Stream-oriented → low memory device

- Buffer oriented approach – high performance servers.

Transaction – AMQP

. NO fragmentation

Long lived messages – don't exist in MQTT

AMQP – classic message queues, round robin, store-and-forward and Combination – 'Filters' are powerful

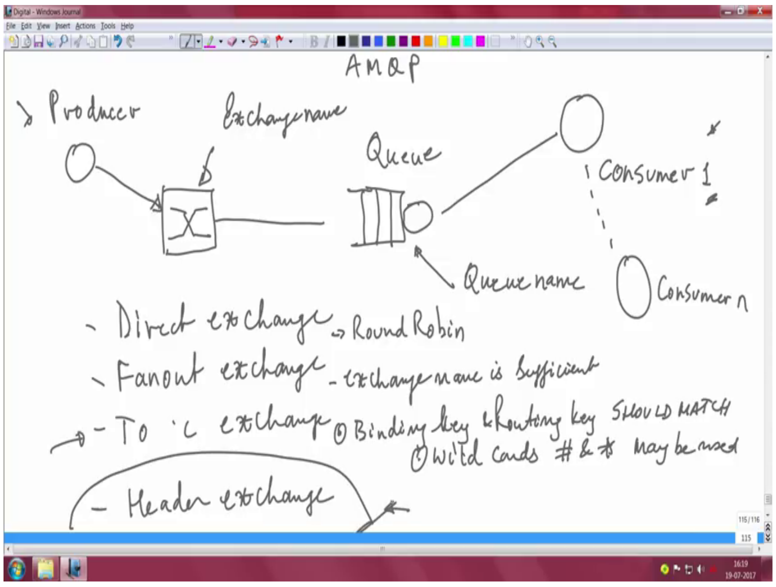
So MQTT I would say is more I would say stream based I would say it is a streaming methodology or rather it uses streaming. So, when you say it is streaming stream oriented it is sort of angering with constrained devices where you have low memory right you have low memory and this low memory client essentially; low memory device is quite amenable to this stream oriented protocol. Whereas, AMQP its buffer it is more like a buffer oriented it is more like a buffer oriented approach which means you can have high performance servers and MQTT you cannot really split a message into two parts, whereas AMQP it say I can take message in sort of allow it to transmit large messages even on constrained environments.

So, MQTT says no fragmentation no fragmentation as against AMQP it says no problem see the trouble with MQTT broker based approach you have a publisher and you have a subscriber you can be offline for a while for a short while, but by and large it assumes that both are online it assumes that both are really online. So, long lived messages long lived messages do not exist do not exist in MQTT. Whereas, again coming back to AMQP this guy is any form of messaging is fine you can be taking about classical classic sorry classic message queues; queues round robin right store and forward and combinations. So, all of this is possible.

So, what this actually means is some customers can get copies of messages while others full staring from the same queue that is possible and the filters are powerful you have filters are powerful I would say and like MQTT. So, you could apply very nice complex rules which form like filters and you can extract data you can pull the data from them. So, that makes it very interesting and very exciting protocol. And AMQP has another advantage that it is it basically likes a transaction it likes to do get into a transaction more is possible in AMQP, whereas with MQTT you do not have this possibility of a transaction we will discuss security and other related the features as we go along in the next class, thank you.

So, let us do a few more demonstrations of AMQP and let us understand all the possible exchange types which are there at least 3-2 other important exchange types which will give you an overall idea of how AMQP can be configured I want you to draw your attention to the picture I have drawn here, where we go back and show you the standard picture of a publish subscribe architecture out in this picture what you see is a producer.

(Refer Slide Time: 32:31)


AMQP
Producer
Exchange name
Queue
Consumer 1
Queue name
Consumer n
- Direct exchange → Round Robin
- Fanout exchange - exchange name is sufficient
- To'c exchange ① Binding key & Routing key SHOULD MATCH
② Wild cards # & * may be used
- Header exchange

Which you can see the nice thing about it that name has been changed to a producer and a consumer right. So, they call them producers and consumers instead of publish subscribe which we know very well in the MQTT paradigm.

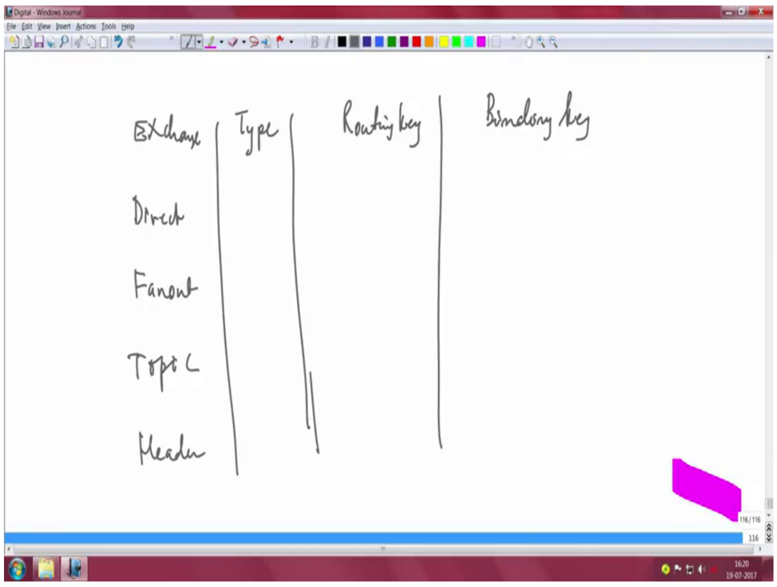
Essentially, producers always give it to what is known as an exchange and there is an exchange name associated with that there is a Q to which this data goes from the exchange and then there is a binding which you specify between the exchange and the queue which is a very critical thing and then there is a queue name as well right and then there are consumer you are essentially IoT devices. So, what are the types of exchanges which are names of different types of exchanges which are possible the type of exchanges you can have a direct exchange you can have fan out exchange you can have topic exchange and you can have header exchange.

Let us go one-by-one we completed direct exchange in the previous demo, but what we would also like to show you is the nice possibility of direct exchange round robin scheduling you can do a round robin kind of data. Let us say you know giving data or for consumers data can be served in a round robin fashion. In other words the first data packet that arise from a producer can go to consumer one and the second data packet from producer can go to consumer number 2 right. So, this is something that allows you we could see that demonstration of round robin.

Now, with related to with relation to the name of exchanges and so on and. So, forth well fan out as I already described to you last time is like a simple thing which is a broadcast based system which essentially means that it does not really matter what you give as an exchange. In fact, you do not even need to specify the queue name it is just sufficient if you specify the name of the exchange because it goes to all consumers moment they are connected to that particular exchange it simply connects to on. So, it is like a one to many broad cast fan out.

Now, the regarding topic exchange this is an interesting thing that you have to note is that the binding key and the routing should actually match otherwise it will not get delivered to the consumers. However, there are possibilities of using the wild cards that we know very well and such wildcards include the hash. And the star which is something you can actually exploit for when you set up the whole AMQP architecture header exchange is something I want you to explore all by yourself it is also an interesting thing and so we will not be able to spend much time doing a header exchange.

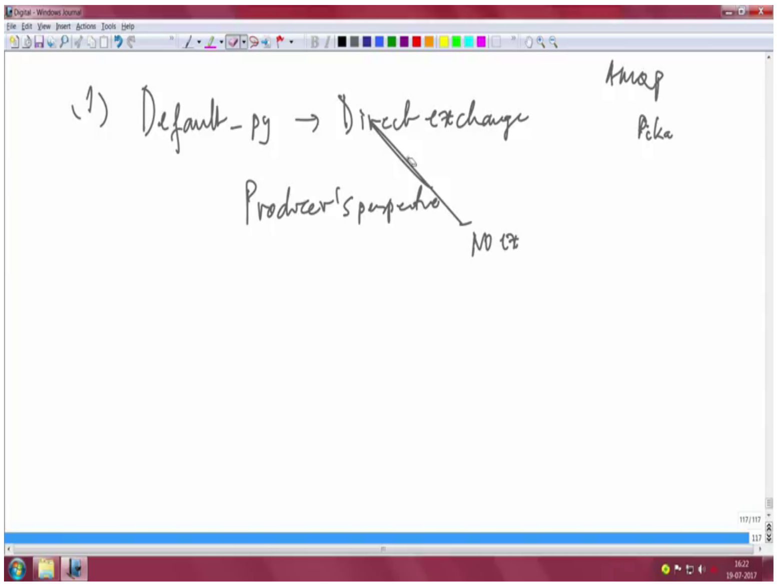
(Refer Slide Time: 36:01)



So, now, let us move on to understand from a from code perspective how exactly these things actually are specified.

So, essentially you have to keep the name of the exchange direct fan out topic header in mind you have a type you have the routing key and the binding. So, if you fill up this table then you are more or less done right. So, that is the key now let us first start by the by showing you first script that I would like to show you.

(Refer Slide Time: 36:35)



1) Default_py → Direct exchange

Producer's perspective

No Ex

AMQP

Pika

In fact, Abhirami and Tejaswini are right here they will demonstrate default underscore p y right this is the name of the file and they will show you something with respect to the direct exchange it is good to know what goes inside this inside this a python script file I must tell you that the AMQP producer consumer scored is Pika we have used Pika I think you can pick it up from the web its and I think in open source. So, you can just take Pika p i k a. So, go to Pika and you can use the scripts. So, direct exchange.

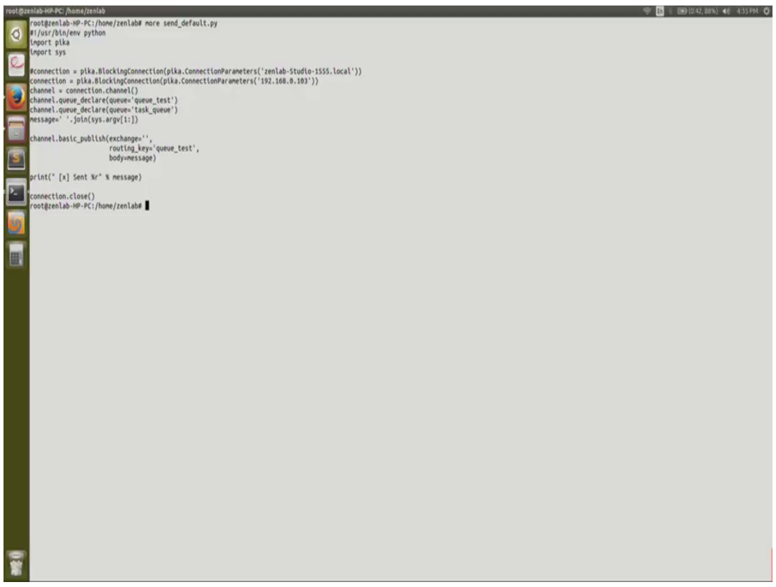
Let me recap from a producers perspective from a producers perspective you do not need to specify the name of any exchange no exchange is required all right. So, I think we will just go and see the demonstration and I will it perhaps mention to you as we do; as we go through the demonstration what are the key things to look out in those particular python script.

So, let us start by showing you some scripts on the producer side. So, so let us take the first script which is on the producer side and. So, let me move this. So, this is the producer this is the producer what you see in the middle is the consumer the rabbit broker which has also support for the exchange and another host which essentially is able to which is basically a consumer you can have multiple instances of the consumer let us focus right now on the producer side.

So, I will ask Tejaswini to type in the default dot p y which is essentially the direct exchange let us open that file. So, now, what will do is we will start looking up at all the scripts from the producers side one by one we will go and I will explain to you the important things in these script use Pika. As I mentioned to you open those scripts and modify them accordingly you should be able to set up a quite simply the AMQP both producer rabbit m Q as the broker which also has the exchange and consumers. Remember that AMQP is basically customer driven not any company driven system.

So, it is a much more powerful much more scalable and robust publish subscribe architecture not just limited to publish subscribe, but also about routing which I mentioned to you right. So, let us move on Default.

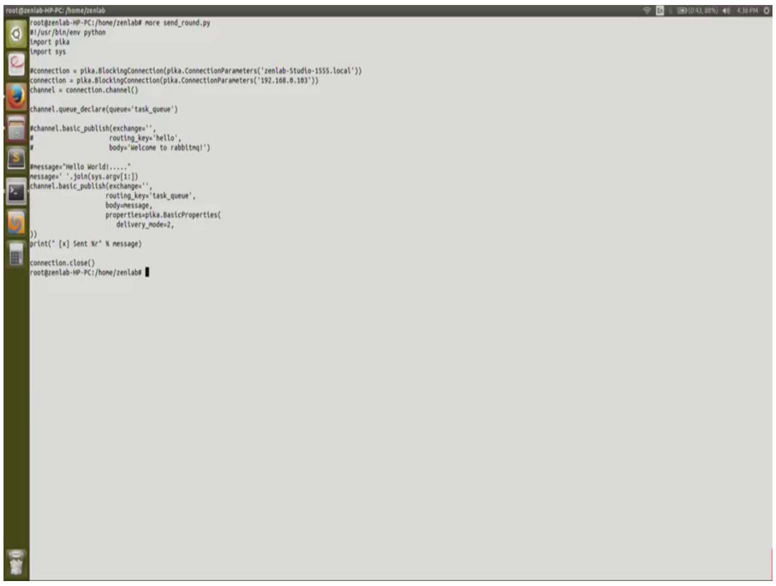
(Refer Slide Time: 39:43)



```
root@zenlab-HP-PC: /home/zenlab
root@zenlab-HP-PC:/home/zenlab# more send_default.py
#!/usr/bin/env python
import pika
import sys

#connection = pika.BlockingConnection(pika.ConnectionParameters('zenlab-Studio-1555.local'))
connection = pika.BlockingConnection(pika.ConnectionParameters('192.168.0.103'))
channel = connection.channel()
channel.queue_declare(queue='queue_test')
channel.queue_declare(queue='task_queue')
message=' '.join(sys.argv[1:])

channel.basic_publish(exchange='',
                      routing_key='queue_test',
                      body=message)

print(" [x] Sent %r" % message)

connection.close()
root@zenlab-HP-PC:/home/zenlab# 
```

So, we will first show you the direct exchange. So, let us open direct exchange direct exchange is you can see is default dot p y here what you should look out is that you have we have not actually specified any exchange name right, but a queue name has been specified. In fact, you can see that it is called queue test and in fact, this whatever is being published can also be published to test underscore queue which is also shown there, but never mind if you have a consumer which connects to queue test he will pick data from queue test, but producer says I want to give it to both the queues both queue test as well as top.

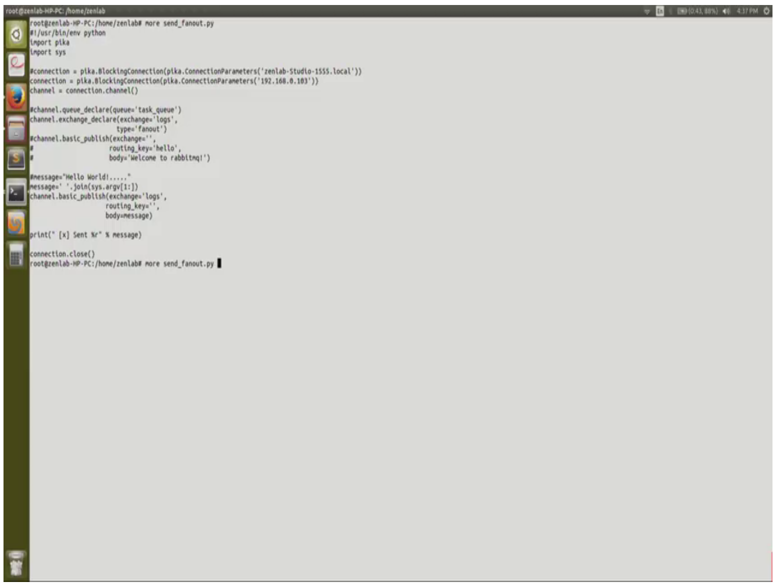
(Refer Slide Time: 40:31)



```
root@zenlab-HP-PC: /home/zenlab
root@zenlab-HP-PC:/home/zenlab# more send_round.py
#!/usr/bin/env python
import pika
import sys

#connection = pika.BlockingConnection(pika.ConnectionParameters('zenlab-Studio-1555.local'))
connection = pika.BlockingConnection(pika.ConnectionParameters('192.168.0.103'))
channel = connection.channel()

channel.queue_declare(queue='task_queue')

#channel.basic_publish(exchange='',
#                      routing_key='hello',
#                      body='Welcome to rabbitmq!')

#message="Hello World!....."
message=' '.join(sys.argv[1:])
channel.basic_publish(exchange='',
                     routing_key='task_queue',
                     body=message,
                     properties=pika.BasicProperties(
                        delivery_mode=2,
))
print(" [x] Sent %r" % message)

connection.close()
root@zenlab-HP-PC:/home/zenlab# 
```

So, that is about the default dot p y let us clear this screen and let us try the round robin part of the direct exchange. Now we go for round robin open again you can see that no name of the exchange specified the queue name is specified here which is task queue. And what is important is that delivery underscore mode which has been set to two this means that indeed it is round robin.

(Refer Slide Time: 41:05)

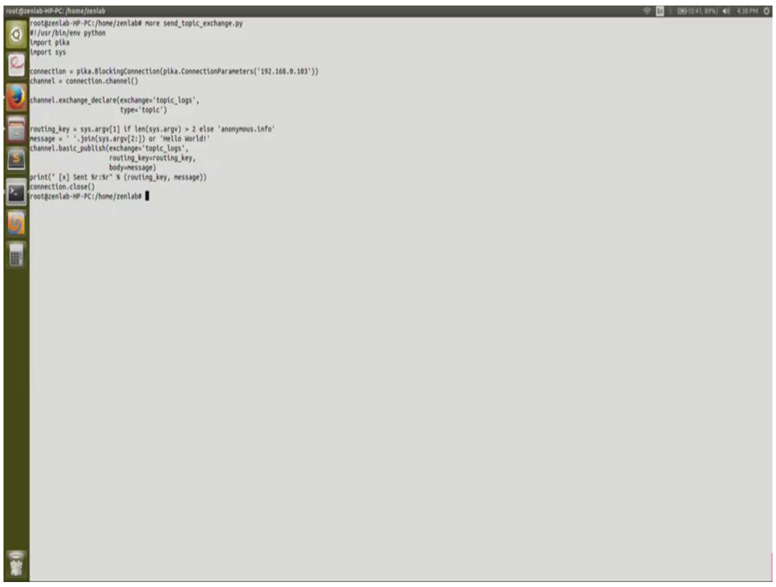


```
root@zenlab-HP-PC: /home/zenlab
root@zenlab-HP-PC:/home/zenlab# more send_fanout.py
#!/usr/bin/env python
import pika
import sys

#connection = pika.BlockingConnection(pika.ConnectionParameters('zenlab-Studio-1555.local'))
connection = pika.BlockingConnection(pika.ConnectionParameters('192.168.0.103'))
channel = connection.channel()

#channel.queue_declare(queue='task_queue')
channel.exchange_declare(exchange='logs',
                         type='fanout')
#channel.basic_publish(exchange='',
#                      routing_key='hello',
#                      body='Welcome to rabbitmq!')

#message="Hello World!....."
message=' '.join(sys.argv[1:])
channel.basic_publish(exchange='logs',
                      routing_key='',
                      body=message)

print(" [x] Sent %r" % message)

connection.close()
root@zenlab-HP-PC:/home/zenlab# more send_fanout.py 
```

Now we clear the screen again and we go and show you complete the thing on fan out in fan out look out for the name of the exchange you can see type is equal to fan out is mentioned recall the table I put in the slide where I want you to fill up. So, essentially type is equal to fan out means it is the name of the exchange and you do not need to specify any routing key only the queue name is also specified and the queue name here is task underscore queue. So, that is the name of the queue.

So, now let us clear again. And now open up topic exchange right here again.

(Refer Slide Time: 41:47)



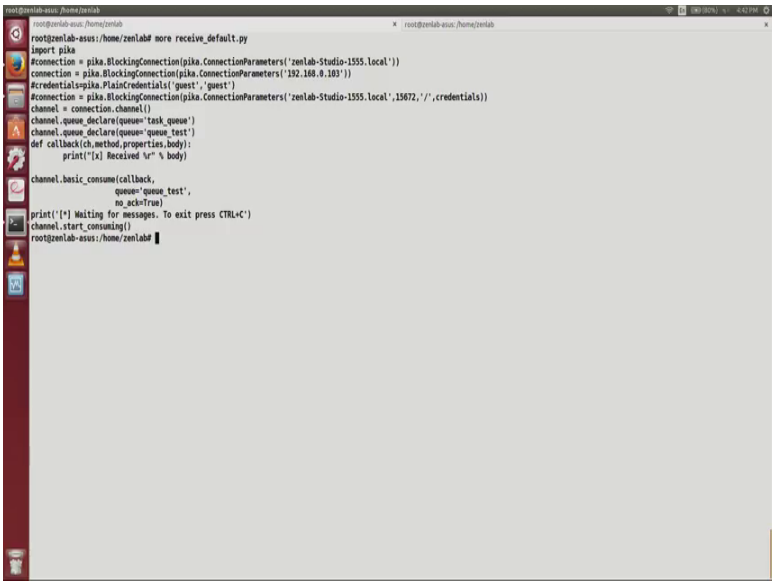
```
root@zenlab-HP-PC:/home/zenlab# more send_topic_exchange.py
#!/usr/bin/env python
import pika
import sys

connection = pika.BlockingConnection(pika.ConnectionParameters('192.168.0.103'))
channel = connection.channel()

channel.exchange_declare(exchange='topic_logs',
                         type='topic')

routing_key = sys.argv[1] if len(sys.argv) > 2 else 'anonymous.info'
message = ' '.join(sys.argv[2:]) or 'Hello World!'
channel.basic_publish(exchange='topic_logs',
                      routing_key=routing_key,
                      body=message)
print(" [x] Sent %r:%r" % (routing_key, message))
connection.close()
root@zenlab-HP-PC:/home/zenlab# 
```

What is important is the exchange name is required. So, you can see the name of the exchange here is topic underscore logs right and this is indeed a type which is topic exchange. So, that is also indicated quite clearly there you do need a routing key right. So, you can see that the routing has been specified as routing key is equal to I am sorry the routing key is specified here as a routing key is equal to underscore routing key which means binding key which will come as an argument while we type will be taken and that will be given to the routing key.

So, the binding key and the routing key have to match essentially that is the key requirement in the topic in the topic type of exchange all right. So, now, these are the most important thing here and this completes the type of the number of scripts that we look up at the producer side.

Now, let us shift and see what actually happens at the consumer side all right now.

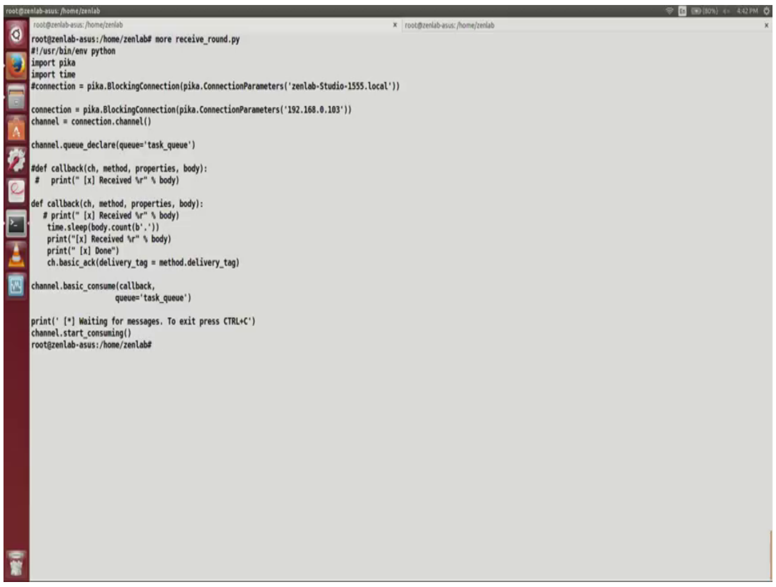
(Refer Slide Time: 43:20)



```
root@zenlab-asus:/home/zenlab# more receive_default.py
import pika
#connection = pika.BlockingConnection(pika.ConnectionParameters('zenlab-Studio-1555.local'))
connection = pika.BlockingConnection(pika.ConnectionParameters('192.168.0.103'))
#credentials=pika.PlainCredentials('guest','guest')
#connection = pika.BlockingConnection(pika.ConnectionParameters('zenlab-Studio-1555.local',15672,'/',credentials))
channel = connection.channel()
channel.queue_declare(queue='task_queue')
channel.queue_declare(queue='queue_test')
def callback(ch,method,properties,body):
        print("[x] Received %r" % body)

channel.basic_consume(callback,
                      queue='queue_test',
                      no_ack=True)
print('[*] Waiting for messages. To exit press CTRL+C')
channel.start_consuming()
root@zenlab-asus:/home/zenlab# 
```

This is the screen which is on the consumers side we will start by looking up at the direct exchange which is given by the script name receive underscore default dot p y. Let us open that now, Abhirami shows that what is very important is the queue name which has to be a specified and just you have to look out for that the rest of it is as a standard thing.

(Refer Slide Time: 43:54)

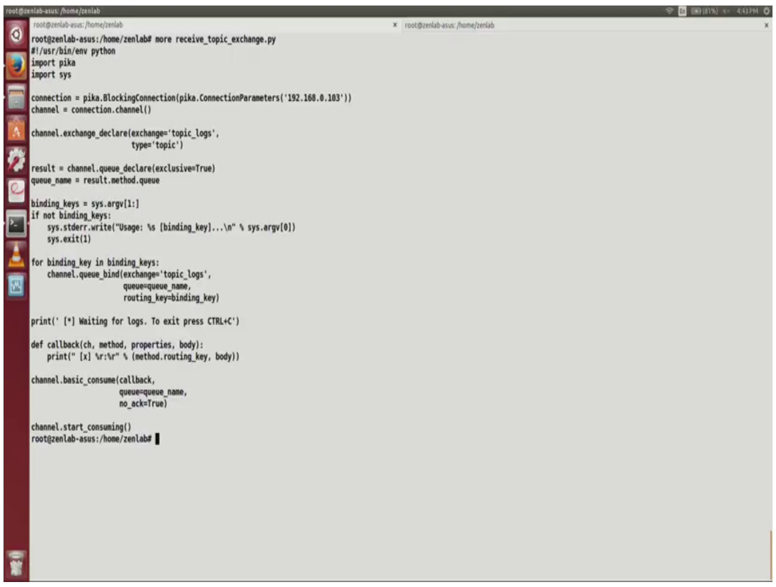


```
root@zenlab-asus:/home/zenlab# more receive_round.py
#!/usr/bin/env python
import pika
import time
#connection = pika.BlockingConnection(pika.ConnectionParameters('zenlab-Studio-1555.local'))

connection = pika.BlockingConnection(pika.ConnectionParameters('192.168.0.103'))
channel = connection.channel()

channel.queue_declare(queue='task_queue')

#def callback(ch, method, properties, body):
 #   print(" [x] Received %r" % body)

def callback(ch, method, properties, body):
   # print(" [x] Received %r" % body)
    time.sleep(body.count(b'.'))
    print("[x] Received %r" % body)
    print(" [x] Done")
    ch.basic_ack(delivery_tag = method.delivery_tag)

channel.basic_consume(callback,
                      queue='task_queue')

print(' [*] Waiting for messages. To exit press CTRL+C')
channel.start_consuming()
root@zenlab-asus:/home/zenlab#
```

So, let us clear this and now let us move on to see the same direct exchange, but this time round robin again you should look out for the queue name right which you can see there is channel underscore dot queue underscore declare queue is equal to task underscore queue that indeed is the queue name right very good. So, that is about direct exchange now let us do clear screen and let us try fan out.

In fan out what you should look for is the exchange name and you should also look out for the type which is which is equal to the fan out the name of the exchange is logs as you can see there and the type is fan out. Now finally, I will do a clear screen again.

(Refer Slide Time: 44:45)



```
root@zenlab-asus:/home/zenlab# more receive_topic_exchange.py
#!/usr/bin/env python
import pika
import sys

connection = pika.BlockingConnection(pika.ConnectionParameters('192.168.0.103'))
channel = connection.channel()

channel.exchange_declare(exchange='topic_logs',
                         type='topic')

result = channel.queue_declare(exclusive=True)
queue_name = result.method.queue

binding_keys = sys.argv[1:]
if not binding_keys:
    sys.stderr.write("Usage: %s [binding_key]...\n" % sys.argv[0])
    sys.exit(1)

for binding_key in binding_keys:
    channel.queue_bind(exchange='topic_logs',
                       queue=queue_name,
                       routing_key=binding_key)

print(' [*] Waiting for logs. To exit press CTRL+C')

def callback(ch, method, properties, body):
    print(" [x] %r:%r" % (method.routing_key, body))

channel.basic_consume(callback,
                      queue=queue_name,
                      no_ack=True)

channel.start_consuming()
root@zenlab-asus:/home/zenlab# 
```

And then we will open the topic exchange able to open a file which is called topic exchange here what you should look for is the exchange name which is the topic underscore logs and you should look at that type which is actually the exchange type is called a topic what you should really look out for, which we actually mentioned at the producer side when Tejaswini was showing the demonstration.

Abhirami points us to another important line in the in this consumer side which corresponds to the routing key equal to the binding key. This is the point really that you have to specify at the consumer side which is quite obvious right and that becomes the binding key becomes an argument. When the producer is actually pushing out data and that is how producers and consumers are connected via the exchange and the binding between exchange and the queues which are associated now what we will do is we will quickly.

So, this is mainly what we can show you in terms of scripts, but a real demonstration would mean that you should actually practice by yourself and see how exactly it happens, because it has to pass through the broker.