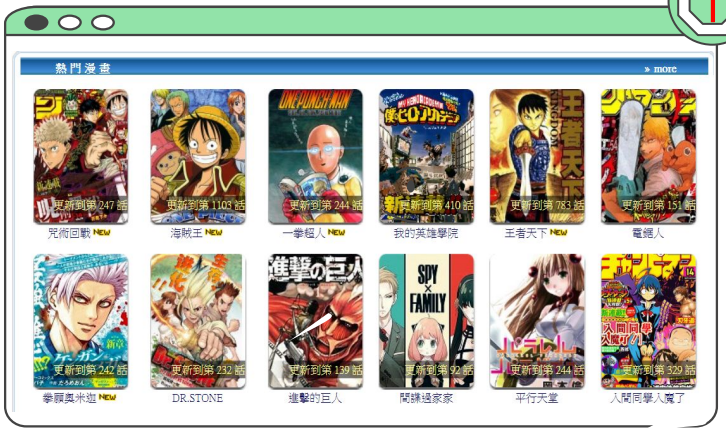
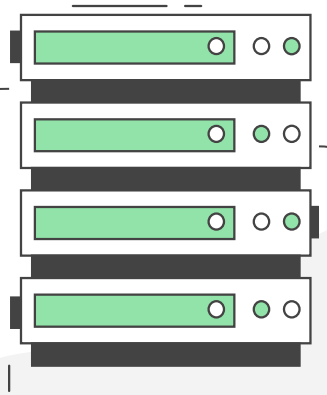
Qpid-Proton實作

Apache QPID





1

2

3

111971013 楊昇豐
111971017 許瑋如

大綱

01 **目標** 情境

02 **步驟**拆解 暨 **虛擬碼**示意

03 **步驟**拆解 暨 **QPID** 頁面對照

04 遭逢 **困難**

05 實際 **演示**

01 目標情境

根據【選取的名冊】，爬取熱門漫畫的最新一話

(1) 漫畫示意：

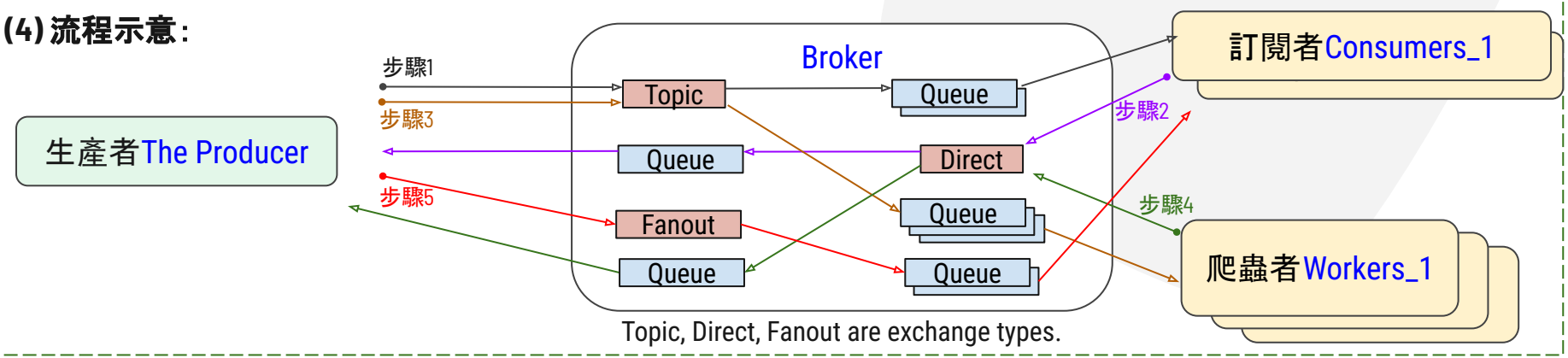
(2) 物件：

- Producer * 1
- Broker * 1
- Consumers * 2
  Workers * 3

(3) 步驟：

1. 擷取當日熱門漫畫名冊，按**Topic** 傳送給訂閱者。
2. 選取名冊，按**Direct**回傳指定漫畫給生產者。
3. 準備urls，按**Topic**傳送給爬蟲者，爬取儲存在本機。
4. 按**Direct**回傳工作完成給生產者。
5. 按**Fanout**傳送工作完成給訂閱者。

(4) 流程示意：



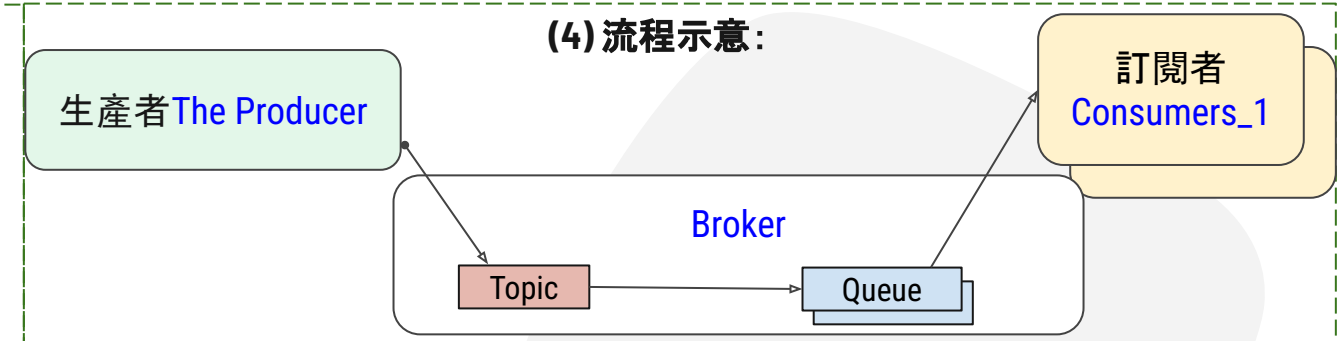
02 步驟拆解 暨 虛擬碼示意

根據【選取的名冊】，爬取熱門漫畫的最新一話

(3) 步驟：

1. 擷取當日熱門漫畫名冊，按**Topic**傳送給訂閱者。
2. 選取名冊，按**Direct**回傳指定漫畫給生產者。
3. 準備urls，按**Topic**傳送給爬蟲者，爬取儲存在本機。
4. 按**Direct**回傳工作完成給生產者。
5. 按**Fanout**傳送工作完成給訂閱者。

(4) 流程示意：



情境	步驟1	步驟2	步驟3	步驟4	步驟5
TheProducer	DeliverHotComicToday				
Consumers	WaitForComicList		--		
Workers	--				

步驟1虛擬碼

```
hotComicTodayJson = httpClient.getHotComicList()
sender = container.create_sender(conn, "amq.topic/the_producer.hot_comic_today")
sender.send(Message(body=hotComicTodayJson )
                , properties={'TheProducerSent': "yes", 'HotComicToday': "yes" }
                , id = "TheProducer")
```

```
receiver = container.create_receiver(conn, "amq.topic/the_producer.hot_comic_today"
            , options=Selector("TheProducerSent = 'yes' AND HotComicToday = 'yes'") )
```

步驟1命令列截圖

```
<<<<<<< TheProducer on_sendable begins : DeliverHotComicToday
>>>>>>> TheProducer on_sendable done
```

```
<<<<<<< Consumers_1 on_message begins: WaitForComicList
Received comics today :
{
  "1": {
    "comic": "咒術回戰",
    "status": "第247話"
  },
  "2": {
    "comic": "海賊王",
    "status": "第1103話"
  },
  "3": {
```

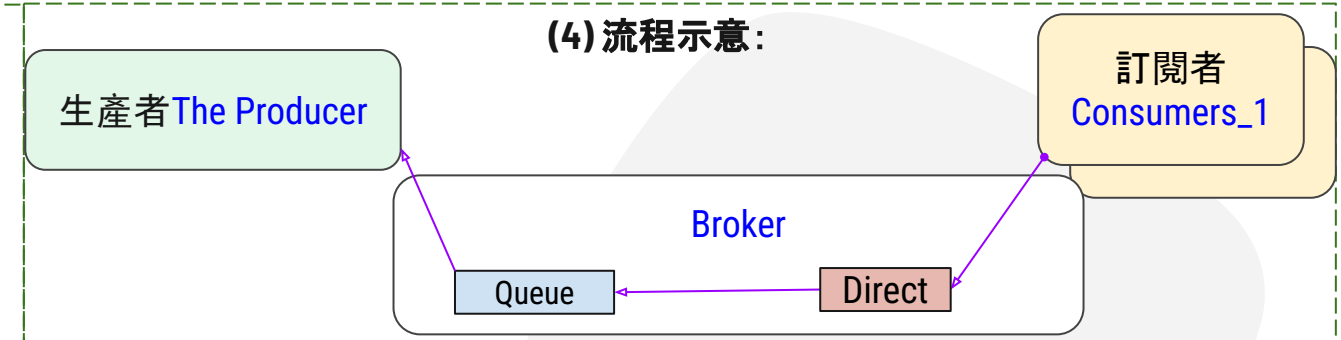
02 步驟拆解 暨 虛擬碼示意

根據【選取的名冊】，爬取熱門漫畫的最新一話

(3) 步驟：

1. 擷取當日熱門漫畫名冊，按**Topic** 傳送給訂閱者。
2. 選取名冊，按**Direct**回傳指定漫畫給生產者。
3. 準備urls，按**Topic**傳送給爬蟲者，爬取儲存在本機。
4. 按**Direct**回傳工作完成給生產者。
5. 按**Fanout**傳送工作完成給訂閱者。

(4) 流程示意：



情境	步驟1	步驟2	步驟3	步驟4	步驟5
TheProducer	DeliverHotComicToday	ReceiveComicListChosen			
Consumers	WaitForComicList		--		
Workers	--				

步驟2虛擬碼

```
receiver = container.create_receiver(conn, "amq.direct/consumers_reply")

sender = container.create_sender(conn, "amq.direct/consumers_reply")
sender.send(Message(body=HotComicChosen,  id = “Consumers_1”))
```

步驟2命令列截圖

```
<<<<<<< TheProducer on_message, on_sendable begins : ReceiveComicListChosen
Received msg from Consumers_1
Consumers_1 choose : {
  "comic": {
    "ordinal": "2",
    "name": {
      "comic": "海賊王",
      "status": "第1103話"
    }
  }
}
```

```
<<<<<<< Consumers_1 on_message begins: WaitForComicList
Received comics today :
{
  "1": {
    "comic": "咒術回戰",
    "status": "第247話"
  },
  "2": {
    "comic": "海賊王",
    "status": "第1103話"
  },
  "3": {
2
>>>>>>>Consumers_1 on_message done
```

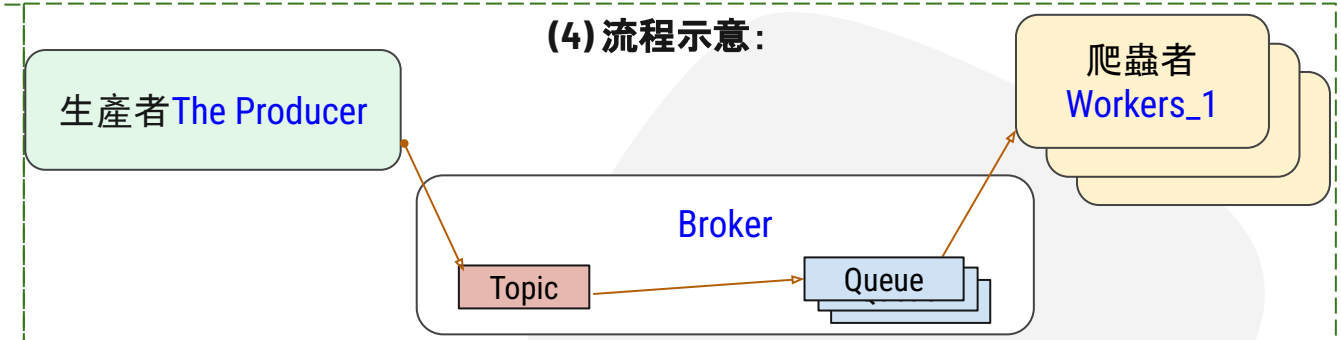
02 步驟拆解 暨 虛擬碼示意

根據【選取的名冊】，爬取熱門漫畫的最新一話

(3) 步驟：

1. 擷取當日熱門漫畫名冊，按**Topic** 傳送給訂閱者。
2. 選取名冊，按**Direct**回傳指定漫畫給生產者。
3. 準備urls，按**Topic**傳送給爬蟲者，爬取儲存在本機。
4. 按**Direct**回傳工作完成給生產者。
5. 按**Fanout**傳送工作完成給訂閱者。

(4) 流程示意：



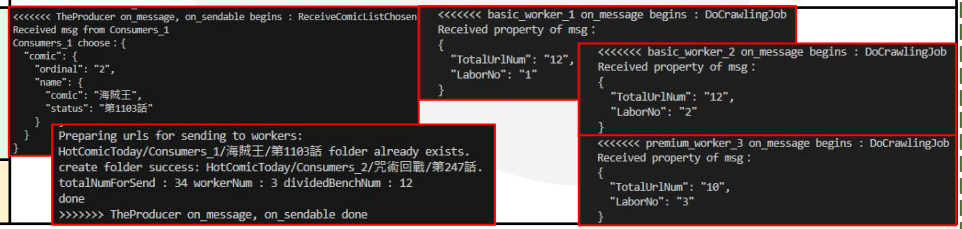
情境	步驟1	步驟2	步驟3	步驟4	步驟5
TheProducer	DeliverHotComicToday		ReceiveComicListChosen		
Consumers		WaitForComicList		--	
Workers	--			DoCrawlingJob	

步驟3虛擬碼

```
sender =  container.create_sender(conn, "amq.topic/the_producer.crawling_list")
totalNumForSend = len(UrlsTupleList)----------------------------------------urls數量
workerNum = 2 if totalNumForSend <= 20 else 3-------------------------------隨需伸縮
dividedBenchNum = round(totalNumForSend / workerNum)------------------------裁切List的基準
sender.send(Message(body=UrlsTupleList[0:dividedBenchNum])
                , properties={'TotalUrlNum': "12", 'LaborNo': "1" }
                , id = "TheProducer")
```

```
receiver = container.create_receiver(conn, "amq.topic/the_producer.crawling_list"
                                     , options=Selector("LaborNo = '1'"))
```

步驟3命令列截圖



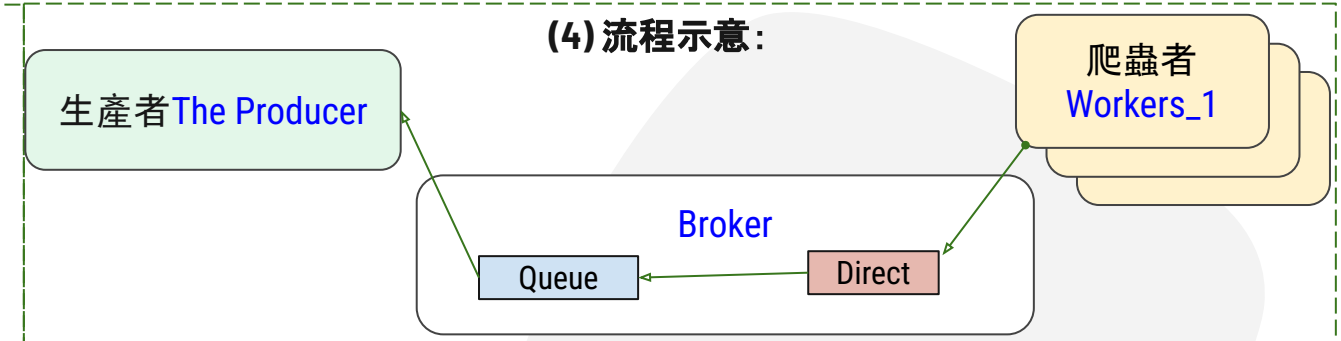
02 步驟拆解 暨 虛擬碼示意

根據【選取的名冊】，爬取熱門漫畫的最新一話

(3) 步驟：

1. 擷取當日熱門漫畫名冊，按**Topic** 傳送給訂閱者。
2. 選取名冊，按**Direct**回傳指定漫畫給生產者。
3. 準備urls，按**Topic**傳送給爬蟲者，爬取儲存在本機。
4. 按**Direct**回傳工作完成給生產者。
5. 按**Fanout**傳送工作完成給訂閱者。

(4) 流程示意：



情境	步驟1	步驟2	步驟3	步驟4	步驟5
TheProducer	DeliverHotComicToday	ReceiveComicListChosen		ReceiveWorkerCondition	
Consumers	WaitForComicList		--		
Workers	--		DoCrawlingJob		

步驟4虛擬碼

```
receiver = container.create_receiver(conn, "amq.direct/workers_reply")

sender = container.create_sender(conn, "amq.direct/workers_reply")
sender.send(Message(body="done", id = workerId))
```

步驟4命令列截圖

```
<<<<<<< TheProducer on_message, on_sendable begins : ReceiveWorkerCondition
Received premium_worker_3 feedback : done
Received basic_worker_1 feedback : done
Received basic_worker_2 feedback : done
```

```
<<<<<<< basic_worker_2 on_message begins : DoCrawlingJob
Received property of msg :
{
  "TotalUrlNum": "12",
  "LaborNo": "2"
}
```

```
<<<<<<< basic_worker_1 on_message begins : DoCrawlingJob
Received property of msg :
{
  "TotalUrlNum": "12",
  "LaborNo": "1"
}
>>>>>>> basic_worker_1 on_message done
```

```
<<<<<<< premium_worker_3 on_message begins : DoCrawlingJob
Received property of msg :
{
  "TotalUrlNum": "10",
  "LaborNo": "3"
}
>>>>>>> premium_worker_3 on_message done
```

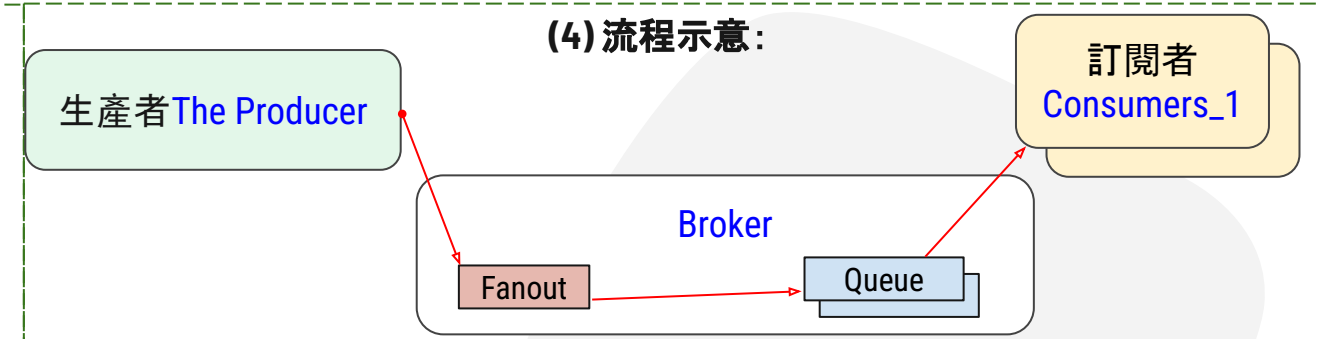
02 步驟拆解 暨 虛擬碼示意

根據【選取的名冊】，爬取熱門漫畫的最新一話

(3) 步驟：

1. 擷取當日熱門漫畫名冊，按**Topic** 傳送給訂閱者。
2. 選取名冊，按**Direct**回傳指定漫畫給生產者。
3. 準備urls，按**Topic**傳送給爬蟲者，爬取儲存在本機。
4. 按**Direct**回傳工作完成給生產者。
5. 按**Fanout**傳送工作完成給訂閱者。

(4) 流程示意：



情境	步驟1	步驟2	步驟3	步驟4	步驟5
TheProducer	DeliverHotComicToday	ReceiveComicListChosen		ReceiveWorkerCondition	
Consumers	WaitForComicList		--		WaitForComplete
Workers	--		DoCrawlingJob		

步驟5虛擬碼

```
sender =  container.create_sender(conn, "amq.fanout")
sender.send(Message(body="Mission Complete!"
                    , properties={'TheProducerSent': "yes", 'HotComicToday': "done" }
                    , id = "TheProducer"))
```

```
self.receiver = event.container.create_receiver(conn, "amq.fanout"
        , options=Selector("TheProducerSent = 'yes' AND HotComicToday = 'done'"))
```

步驟5命令列截圖

```
<<<<<<< TheProducer on_message, on_sendable begins : ReceiveWorkerCondition
Received premium_worker_3 feedback : done
Received basic_worker_1 feedback : done
Received basic_worker_2 feedback : done
>>>>>>> TheProducer on_message, on_sendable done
```

```
<<<<<<< Consumers_1 on_message begins : WaitForComplete
Received comics complete :
Mission Complete!
>>>>>>>Consumers_1 on_message done
```

```
<<<<<<< Consumers_2 on_message begins : WaitForComplete
Received comics complete :
Mission Complete!
>>>>>>>Consumers_2 on_message done
```

03 步驟拆解 暨 QPID 頁面對照

根據【選取的名冊】，爬取熱門漫畫的最新一話

情境	步驟1	步驟2	步驟3	步驟4	步驟5
TheProducer	DeliverHotComicToday		ReceiveComicListChosen	ReceiveWorkerCondition	
Consumers	WaitForComicList		--		WaitForComplete
Workers	--		DoCrawlingJob		

步驟1
步驟3

Broker | Exchange: amq.topic | Exchange: amq.direct | Exchange: amq.fanout | Exchange: amq.match

Bindings

Destination	Binding Key	Arguments
qpidsub_/Consumers__1_/c46c3413-1a02-4dba-b0d6-e4b37ca445ea_/nondurable	the_producer.hot_comic_today	{"x-filter-jms-selector":"TheProducerSent = 'yes' AND HotComicToday = 'yes'"}
qpidsub_/Consumers__2_/3876326d-aebe-4bcd-886b-75ccb811ae27_/nondurable	the_producer.hot_comic_today	{"x-filter-jms-selector":"TheProducerSent = 'yes' AND HotComicToday = 'yes'"}
qpidsub_/premium__worker__3_/b94e4632-62f3-465b-985e-1ba3ef412510_/nondurable	the_producer.crawling_list	{"x-
qpidsub_/basic__worker__2_/83ceb62b-a44c-46d0-8abb-28849b319256_/nondurable	the_producer.crawling_list	{"x-
qpidsub_/basic__worker__1_/3802b49c-fa14-4958-b3f4-033097017f07_/nondurable	the_producer.crawling_list	{"x-

The "binding" defines the relationship between a *message queue* and an *exchange* and provides the message routing criteria.

步驟3
步驟2

Broker | Exchange: amq.topic | Exchange: amq.direct | Exchange: amq.fanout | Exchange: amq.match

Bindings

Destination	Binding Key	Arguments
qpidsub_/TheProducer_/d5ccde79-8c47-4c0d-a135-872c256544f1_/nondurable	workers_reply	{}
qpidsub_/TheProducer_/ddce32fa-1350-47f7-b37f-5aecb6b2ea22_/nondurable	consumers_reply	{}

步驟5

Broker | Exchange: amq.topic | Exchange: amq.direct | Exchange: amq.fanout | Exchange: amq.match

Bindings

Destination	Binding Key	Arguments
qpidsub_/Consumers__1_/dac68707-2f3d-467e-8be7-512aa6e60491_/nondurable	qpidsub_/Consumers__1_/dac68707-2f3d-467e-8be7-512aa6e60491_/nondurable	{"x-filter-jms-selector":"TheProducerSent = 'yes' AND HotComicToday = 'done'"}
qpidsub_/Consumers__2_/9df5b195-50c4-46ad-8a0f-e27297c29b99_/nondurable	qpidsub_/Consumers__2_/9df5b195-50c4-46ad-8a0f-e27297c29b99_/nondurable	{"x-filter-jms-selector":"TheProducerSent = 'yes' AND HotComicToday = 'done'"}

04 遭逢 困難

根據【選取的名冊】，爬取熱門漫畫的最新一話

1. Proton Python Examples，沒有一個範例指出：
   (1) 需要兩個sender傳訊息，該怎麼辦？
   - 如果創多個sender在一個class，將導致同份訊息重複傳送。

Link, redhat

```
class Client(MessagingHandler):
    def __init__(self, url, requests):
        super(Client, self).__init__()
        self.url = url
        self.requests = requests

    def on_start(self, event):
        self.sender = event.container.create_sender(self.url)
```

(2) 需要兩個container時，該怎麼辦？(範例程式都只有一個container)

As the API doc says:
Now that we have defined the logic for handling these events, we create an instance of a ***Container***, *pass it our handler and then enter the event loop by calling run()*. At this point, control passes to the container instance, which will make the appropriate callbacks to any defined handlers.

```
Container(HelloWorld("localhost:5672", "examples")).run()
```

```
waitForComicList = Container(WaitForComicList("localhost:5672"))
waitForComicList.container_id = "Consumers_1"
waitForComplete = Container(WaitForComplete("localhost:5672"))
waitForComplete.container_id = "Consumers_1"
waitForComicList.run()
waitForComplete.run()
```

(3) 想用multiprocessing提升效率，卻沒辦法執行container.run()

```
b = Container(HelloWorld("localhost:5672", "amq.topic/1.2"))
"""
# reduction.py", line 79, in duplicate
# return _winapi.DuplicateHandle(
# PermissionError: [WinError 5] Access is denied
"""
# b_proc = mp.Process(target=bb, args=(b,))
# b_proc.daemon = True
# b_proc.start()
```

```
def bb(PP):
    PP.run()
```

05 實際演示

根據【選取的名冊】，爬取熱門漫畫的最新一話

1

Qpid offers **AMQP 1.0 support** in the following components:
(combine Apache_Qpid_wiki with AMQP Qpid official site)

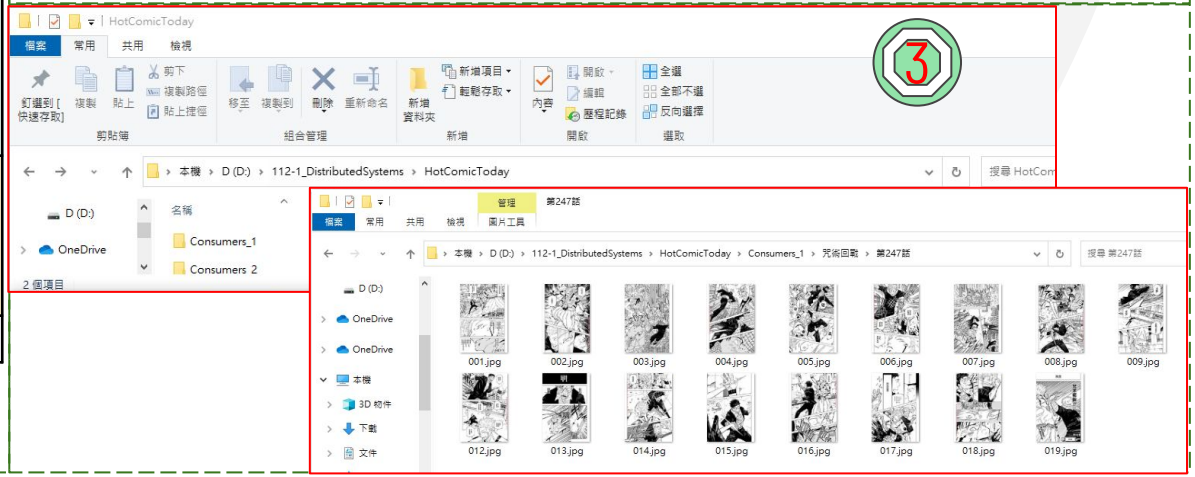
Component	Detailed component	Type
• Qpid Proton	Qpid Proton	Messaging API
	Qpid Proton-J	
• Qpid JMS	Qpid JMS (AMQP 1.0)	
	Qpid JMS AMQP 0-x.	
• Qpid Messaging API	Qpid Messaging API C++	
	Qpid Messaging API Python	
• Broker-J	Broker-J	Messaging server
• C++ broker	C++ Broker	
• Dispatch router	Dispatch router	
	Qpid Interop Test	Messaging tools

vs code 視窗開法

2 目錄 / 程式

			Worker_Basic_1
			Worker_Basic_2
			Worker_Premium_3
The Producer	Consumers_1	Consumers_2	- -

3



05 實際演示

根據【選取的名冊】，爬取熱門漫畫的最新一話

情境	兩個 Worker 爬蟲分工	三個 Worker 爬蟲分工
下載一部動漫	19頁，花費39-45秒	--
下載兩部動漫	34頁，60秒	34頁，花費48秒 86頁，花費93秒
下載相同頁數動漫時，很明顯能看出 Worker 爬蟲分工數量多者花費時間較短。 GitHub：112-1_DistributedSystems_第七組		