Turning the Bazar into an Amazon: Replication, Caching and Consistency

Malak Bawwab

December2020

Contents

1 Project Design

- There are 3 main services:catalog service,order service and front service .
- Both the order and catalog server are replicated - their code and their database files are replicated on multiple machines,front end node is NOT replicated.
- The total number of VMs are 5:3 main services,2 replicas of order and catalog.
- The communication between services is done through Rest APIs and Guzzle http client.

1.1 Cache

- I used the memcache in the front end node,I configured it in the .env file .
- You can do operations with the cache using Cache:: .
- For lookup requests,i stored the bookId(itrmNumber) as a key and bookInformation(quantity,cost,title,topic,id) as a value.
- For search requests,since search requests may return more than 1 book,i stored the topic as a key and array of bookInformation of all books as a value.
- (serverpush techniques)When buy ,update cost,update quantity requests occur,an invalidate request will be sent from backend replicas to front node that contains in-memory cache to remove item from the cache.

1.2 Load Balancing Algorithm

- To deal with the replication, the front end node implement a load balancing algorithm (round-robin) that takes each incoming request and sends it to one of the replicas.
- With round-robin,a client request is forwarded to each server in turn.To do that,i stored 2 separate keys with initial value 1 ,one for catalogLoadBalancing,second for orderLoadBalancing.
- When catalogLoadBalancing=1,forward request to main catalog and set the key to 2.
- When catalogLoadBalancing=2,forward request to replica1-catalog and set the key to 1
- When orderLoadBalancing=1,forward request to main order and set the key to 2.
- When orderLoadBalancing=2,forward request to replica1-order and set the key to 1

1.3 Replication

- The replicas should also use an internal protocol to ensure that any writes to their database are also performed at the other replica to keep them in sync with one another.
- To do that,If any updates occur,a notifyUpdate request will be sent to other replicas containing the new updated value .

2 Database

- I used sqlite database since it is a lightweight database and it is easy to use.
- Front service doesn't use sqlite.
- Catalog service uses sqlite with a table named books,books table contains books information(itemNumber,price,quantity,topic,title).
- Order service uses sqlite with a table named orders, orders table contains a list of all orders received for the books(customerName,vookId,date)

3 Catalog Service

Server ip for the catalog service is 192.168.164.129. Server ip for the replica catalog service is 192.168.164.132.

- **Catalog service** does query operations (by itemNumber ,by topic)and update operations(cost,quantity) and also does some operations that are required to complete buy process which is defined in the order service.
- **GET http://192.168.164.129/search/{topic}** : this request search for a book in books table based on the topic.
- **GET http://192.168.164.129/lookup/{itemNumber}** :this request search for a book based on the itemNumber.
- **PUT http://192.168.164.129/update/book/{itemNumber}/type/{type}/value/{value}** :this request check the type ,if the type is buy(this request is called from order service to complete buy process(decrease quantity by 1).if the type is newValue(set the quantity to the value specified in the request).if the type is increaseNumber(add value to the old quantity).if the type is decreaseNumber(subtract value from the old quantity).
  If the update is successful,this request will send invalidate request to front end that contains cache to remove item from the cache, and also it will send a notifyUpdate request with the new updated value to other catalog replicas that contains the item.

- **PUT http://192.168.164.128/update/book/{itemNumber}/cost/{newCost}**:this request update the price of the book defined by itemNumber.If the update is successful,this request will send invalidate request to front end to remove item from the cache, and also it will send a notifyUpdate request with the new updated value to other catalog replicas that contains the item.
- **GET/notify/{itemNumber}/{updateType}/{newValue}**:
  this request will update the the DB contents to be the same as it in other catalog replicas.UpdateType may be cost(updating cost),(newValue,buy,increaseNumber ,decreaseNumber –updating quantity).

4 Order Service

Server ip for the order service is 192.168.164.133. Server ip for replica the order service is 192.168.164.131.

- Order service does buy operations and getListOfOrders operation.
- **GET http://192.168.164.133/list/orders/{itemNumber}** : this request return a list for all the received orders of the book with this itemNumber.
- **POST http://192.168.164.133/buy/{itemNumber}** :this request send external requests to check if the book exists on the store and is not out of stock(there are items available to buy).if yes,it sends a second request through Guzzle http client to catalog to decrease the quantity of the book by 1(buy operation is successful).
  Also,if the buy operation is successful this request will add an entry in the orders table(customerName,bookId,date).And it will send a notifyUpdate request with the entry information to other orders replicas.
- **GET/notify/{itemNumber}**:
  this request will add the new order on the orders table to be the same as in other order replicas.

5 Front Service

Server ip for the front service is 192.168.164.128. Front services does search,lookup,buy operations by forwarding requests to catalog service(search,lookup) or getting data from cache and order service(buy) through Guzzle http client.I implemented two ways that the client can do:

- **First:**The client can enter these first 2 urls directly in the browser and the POST request using postman:

  - **GET http://192.168.164.128/search/topic/{topic}** : this request check if the cache contains the topic or not.If yes ,it returns the result from cache .If No(topic isn't in the cache),it sends external request to one of the catalog replicas based on the implemented round-robin algorithm and it will then store the result in the cache(key is the topic,value is the array of bookInformation of all returned books).
  - **GET http://192.168.164.128/lookup/number/{itemNumber}** : this request check if the cache contains the itemNumber or not.If yes ,it returns the result from cache .If No(itemNumber isn't in the cache),it sends external request to one of the catalog replicas based on the implemented round-robin algorithm and it will store the result in the Cache(key is the itemNumber,value is the bookInformation of the returned book).
  - **POST http://192.168.164.128/buy/number/{itemNumber}**:it doesn't check cache,the cache is for query requests(lookup/search).It sends external request to one of the order replicas based on the implemented round-robin algorithm .

- **Second:**the user can enter http://192.168.164.128 in the browser,once he entered a GUI(greeting.php page)will appear.

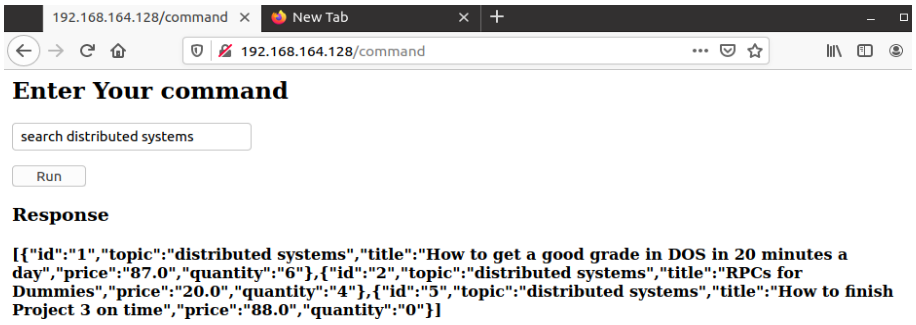
  Greeting page contains a textInput where the user can enter some commands and contains a response section that is resulted from running the entered command.
  If the user doesn't enter any of the 3 commands(search ¡topic¿,lookup ¡itemNumber¿,buy ¡itemNumber¿,the response will be command not found ,else a POST request with a body(containing the full command) will be sent to front server :

  POST http://192.168.164.128 /{command}: this request will parse the entered command ,and based on the command type(search,lookup,buy)it will perform the search,lookup and buy.

6 How to run

- **Run the system as whole:**

  1. Power on the 5 VMs.
     First way:-
     *Open the Front service VM,open the browser and enter http://192.168.164.128
     *Enter the command in the textInput and press run.
     ***Results:**



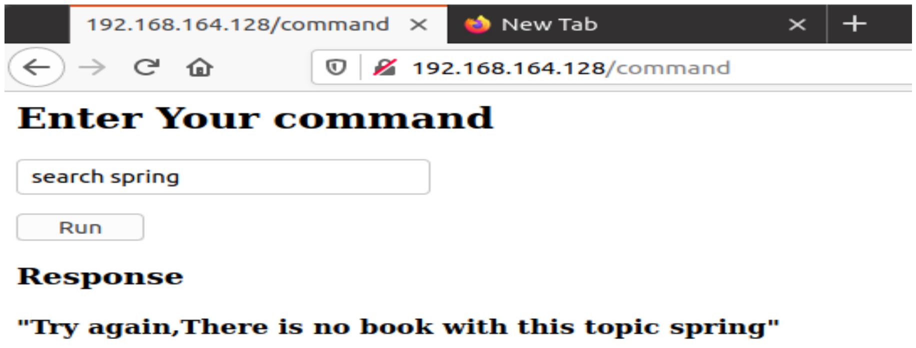
Enter Your command

search distributed systems

Run

Response

```
[{"id":"1","topic":"distributed systems","title":"How to get a good grade in DOS in 20 minutes a day","price":"87.0","quantity":"6"},{"id":"2","topic":"distributed systems","title":"RPCs for Dummies","price":"20.0","quantity":"4"},{"id":"5","topic":"distributed systems","title":"How to finish Project 3 on time","price":"88.0","quantity":"0"}]
```



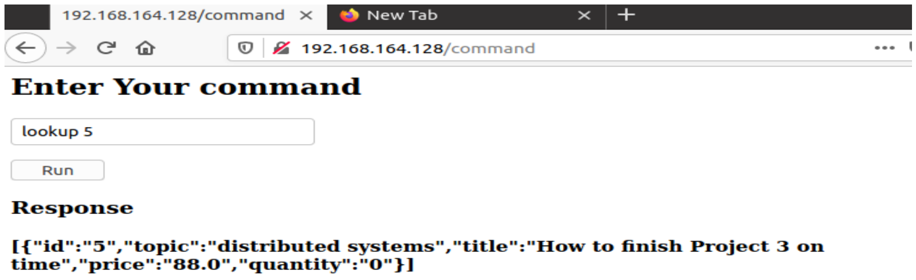
Enter Your command

search spring

Run

Response

```
"Try again,There is no book with this topic spring"
```



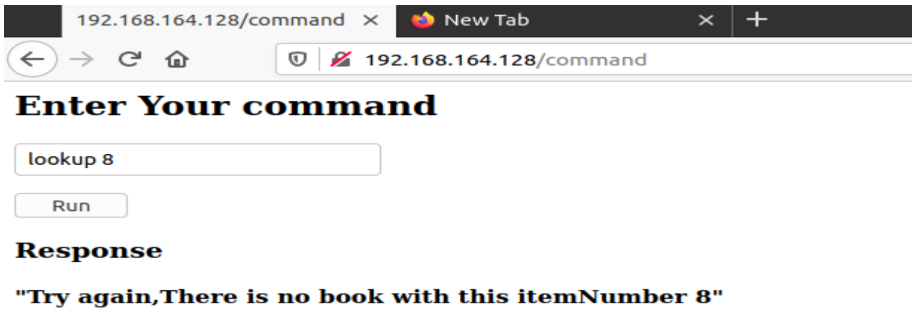
Enter Your command

lookup 5

Run

Response

```
[{"id":"5","topic":"distributed systems","title":"How to finish Project 3 on time","price":"88.0","quantity":"0"}]
```

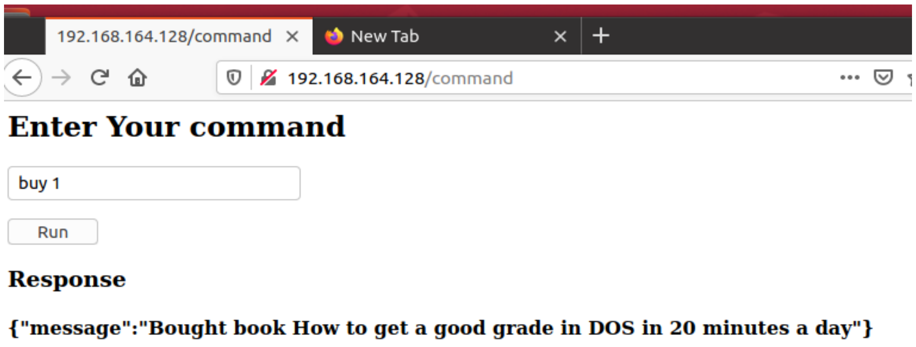

192.168.164.128/command

Enter Your command

lookup 8

Response

"Try again,There is no book with this itemNumber 8"

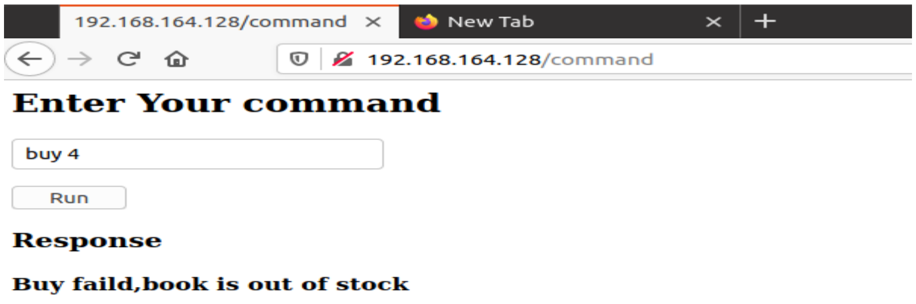

192.168.164.128/command

Enter Your command

buy 1

Response

{"message":"Bought book How to get a good grade in DOS in 20 minutes a day"}

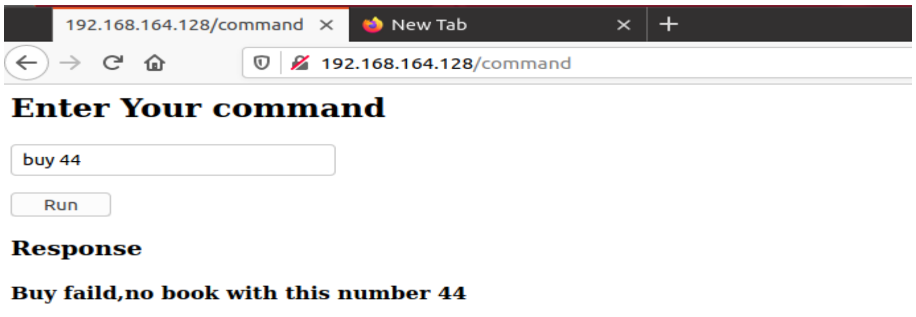

192.168.164.128/command

Enter Your command

buy 4

Response

Buy faild,book is out of stock



192.168.164.128/command

Enter Your command

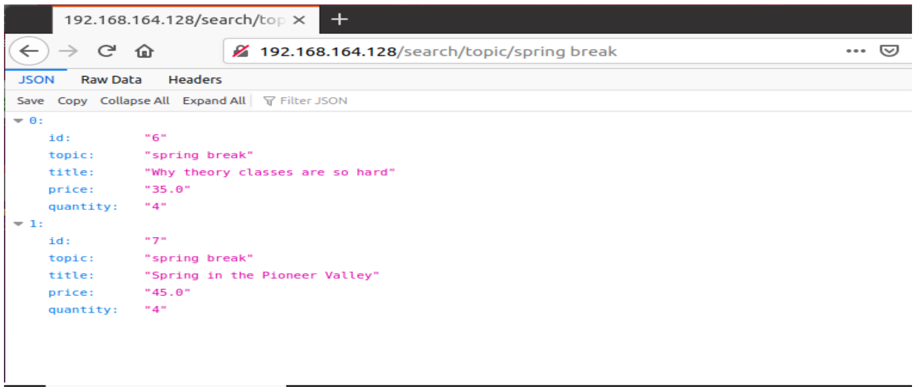
```
buy 44
```

Response

Buy faild,no book with this number 44

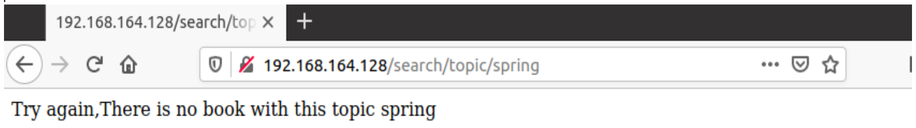
Second way:-

*open the browser and enter http://192.168.164.128/search/topic/{topic}:



192.168.164.128/search/topic/spring break

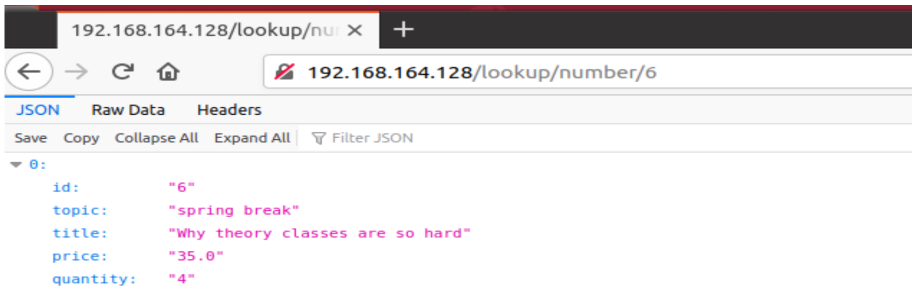
```
0:
    id:         "6"
    topic:      "spring break"
    title:      "Why theory classes are so hard"
    price:      "35.0"
    quantity:   "4"
1:
    id:         "7"
    topic:      "spring break"
    title:      "Spring in the Pioneer Valley"
    price:      "45.0"
    quantity:   "4"
```



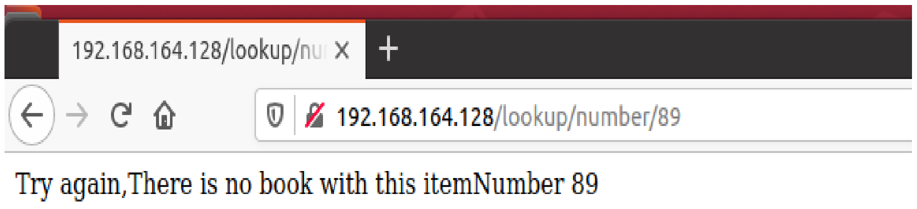
192.168.164.128/search/topic/spring

```
Try again,There is no book with this topic spring
```

*open the browser and enter http://192.168.164.128/lookup/number/{itemNumber}

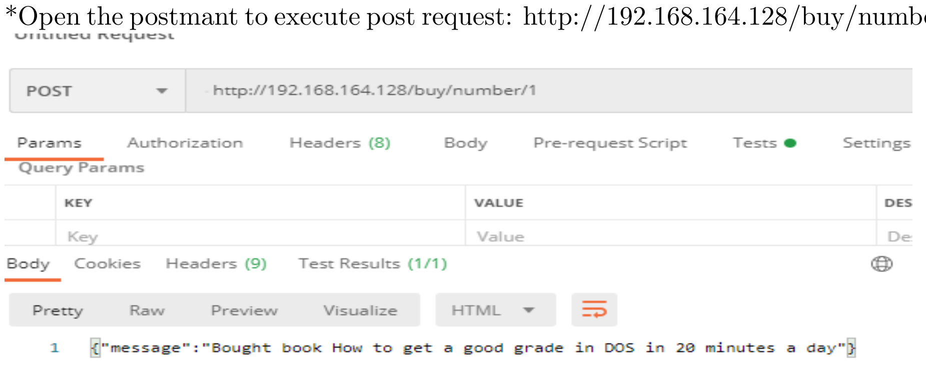

192.168.164.128/lookup/number/6

```
0:
    id:        "6"
    topic:     "spring break"
    title:     "Why theory classes are so hard"
    price:     "35.0"
    quantity:  "4"
```


192.168.164.128/lookup/number/89

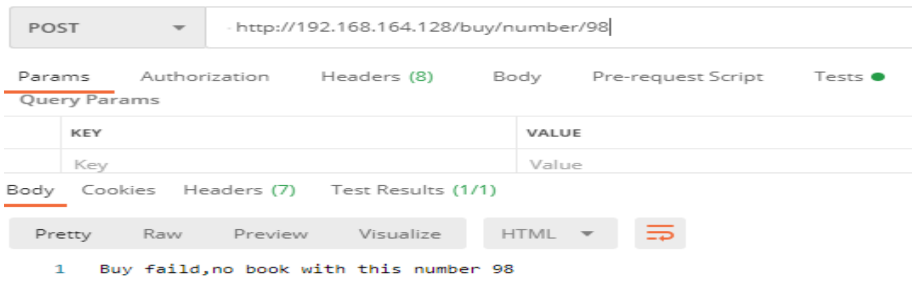
```
Try again,There is no book with this itemNumber 89
```

*Open the postmant to execute post request: http://192.168.164.128/buy/number/{itemNumber}

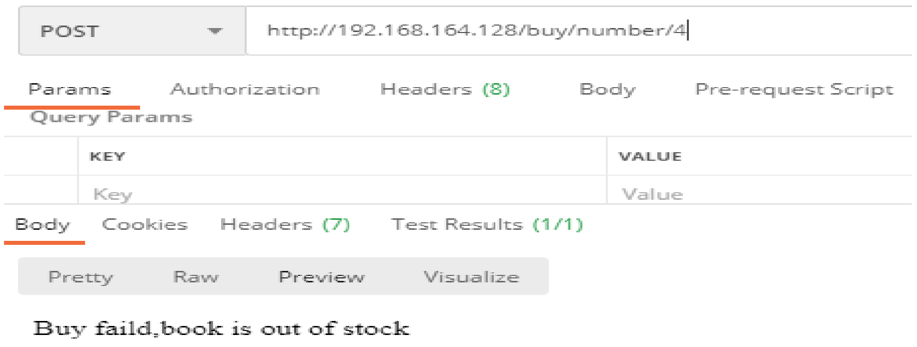

POST http://192.168.164.128/buy/number/1

```
{"message":"Bought book How to get a good grade in DOS in 20 minutes a day"}
```

csm

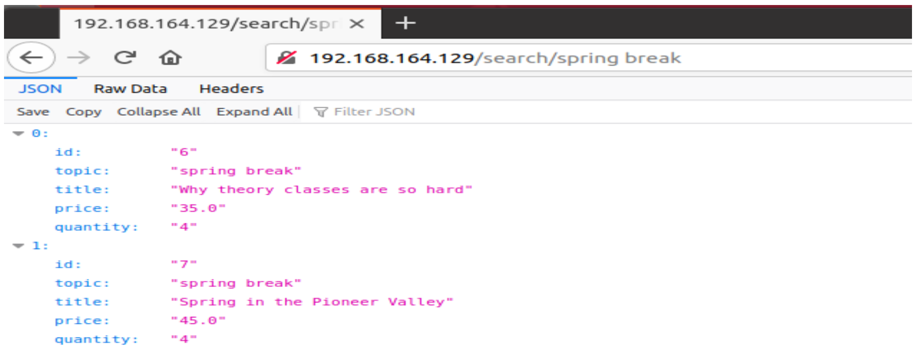

POST http://192.168.164.128/buy/number/98

```
Buy faild,no book with this number 98
```

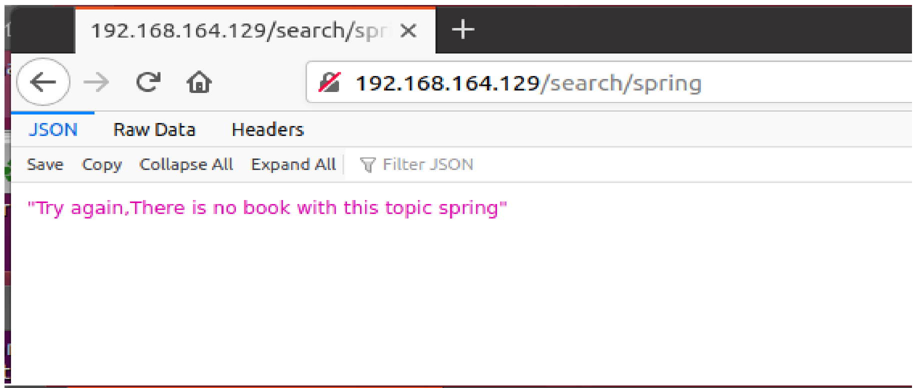

POST http://192.168.164.128/buy/number/4

```
Buy faild,book is out of stock
```

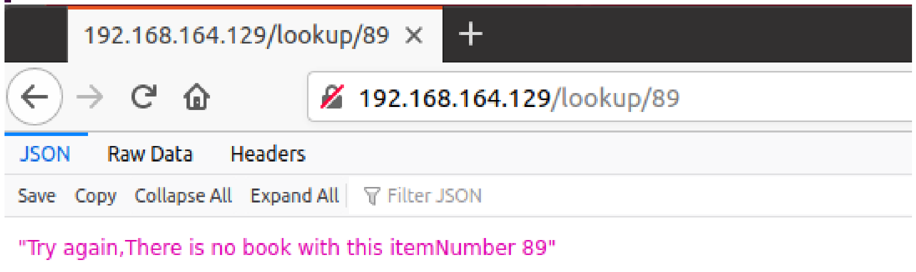
- **Run the catalog service ,replica catalog:**


192.168.164.129/search/spring break

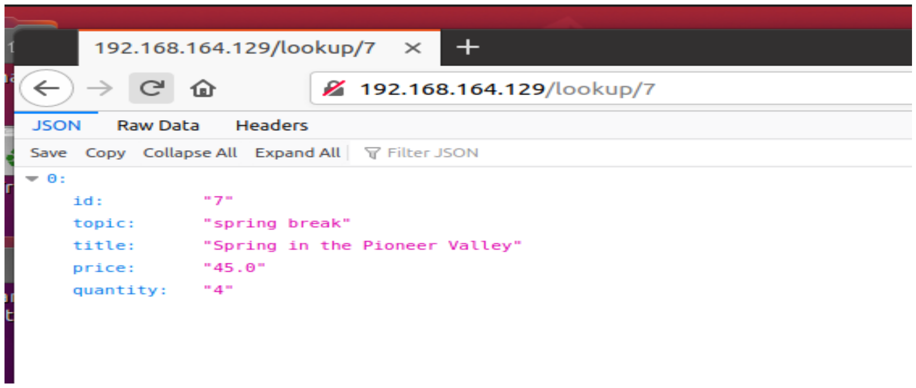
```
0:
    id:         "6"
    topic:      "spring break"
    title:      "Why theory classes are so hard"
    price:      "35.0"
    quantity:   "4"
1:
    id:         "7"
    topic:      "spring break"
    title:      "Spring in the Pioneer Valley"
    price:      "45.0"
    quantity:   "4"
```



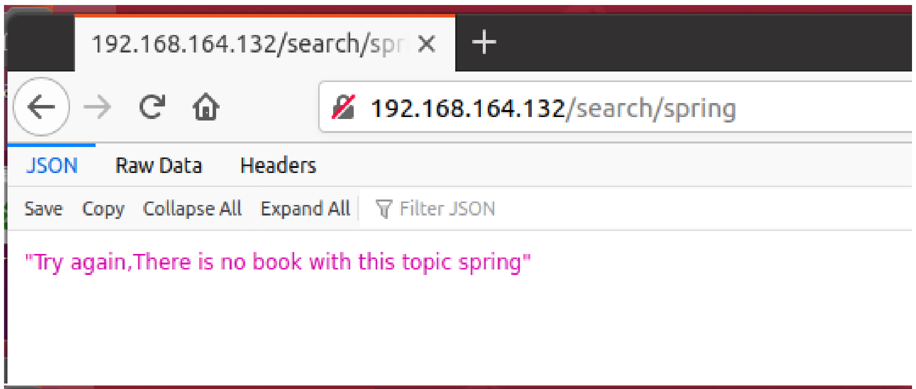
```
"Try again,There is no book with this topic spring"
```



```
"Try again,There is no book with this itemNumber 89"
```

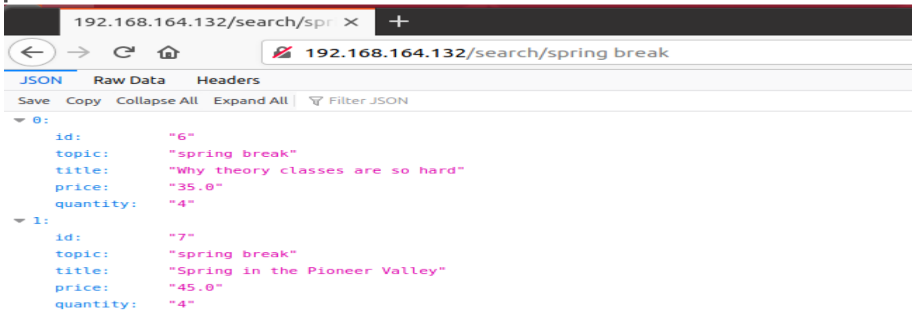


```
▾ 0:
    id:          "7"
    topic:       "spring break"
    title:       "Spring in the Pioneer Valley"
    price:       "45.0"
    quantity:    "4"
```



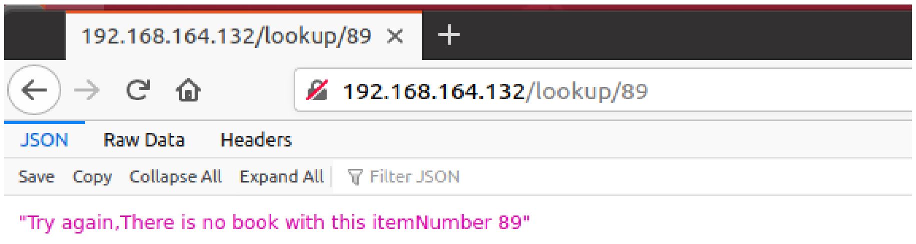
192.168.164.132/search/spring

```
"Try again,There is no book with this topic spring"
```



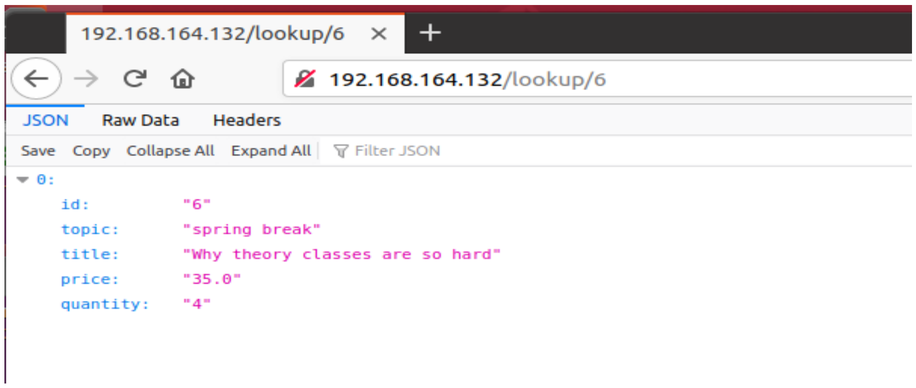
192.168.164.132/search/spring break

```
▼ 0:
    id:         "6"
    topic:      "spring break"
    title:      "Why theory classes are so hard"
    price:      "35.0"
    quantity:   "4"
▼ 1:
    id:         "7"
    topic:      "spring break"
    title:      "Spring in the Pioneer Valley"
    price:      "45.0"
    quantity:   "4"
```



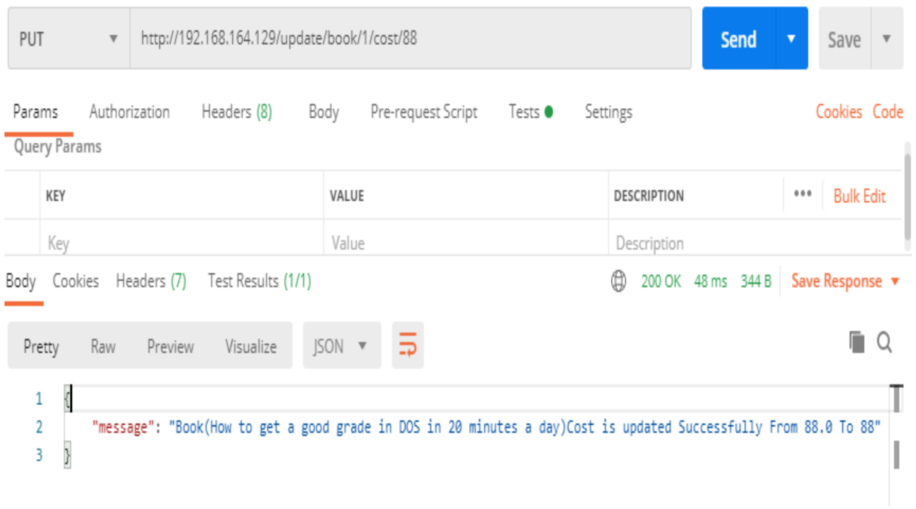
192.168.164.132/lookup/89

```
"Try again,There is no book with this itemNumber 89"
```

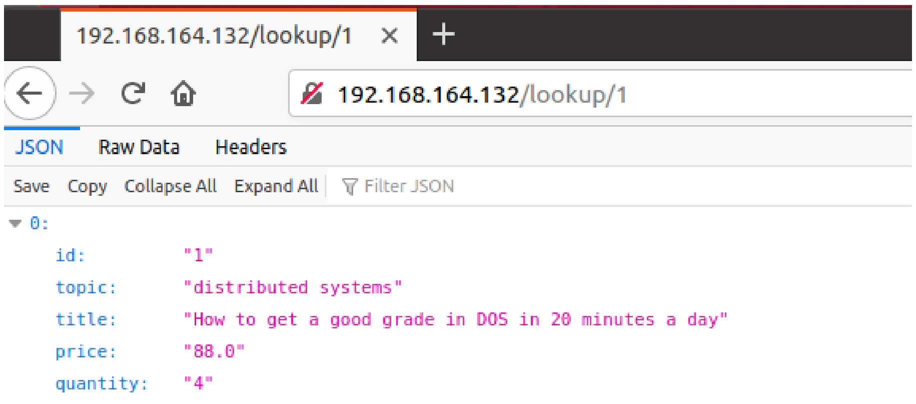


```
0:
    id:         "6"
    topic:      "spring break"
    title:      "Why theory classes are so hard"
    price:      "35.0"
    quantity:   "4"
```

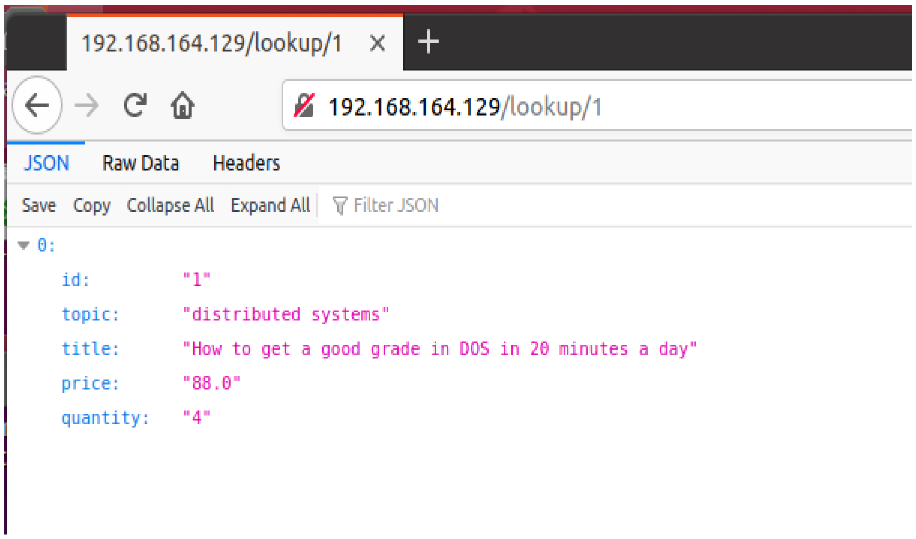
*If i update cost,quantity from any catalog replica(main or replica1),a notify update request will be sent to other catalog replicas.At the end,all replicas will have the same values,DB records.



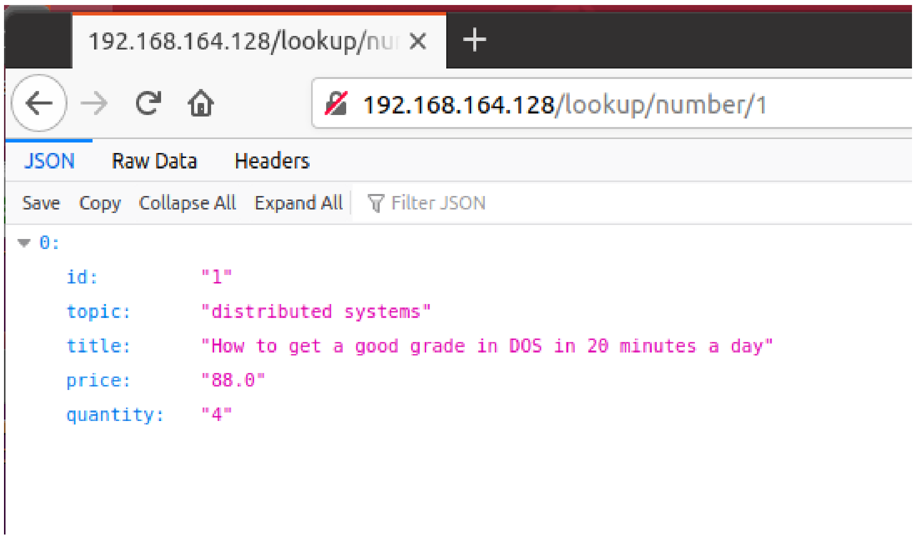
```
{
    "message": "Book(How to get a good grade in DOS in 20 minutes a day)Cost is updated Successfully From 88.0 To 88"
}
```



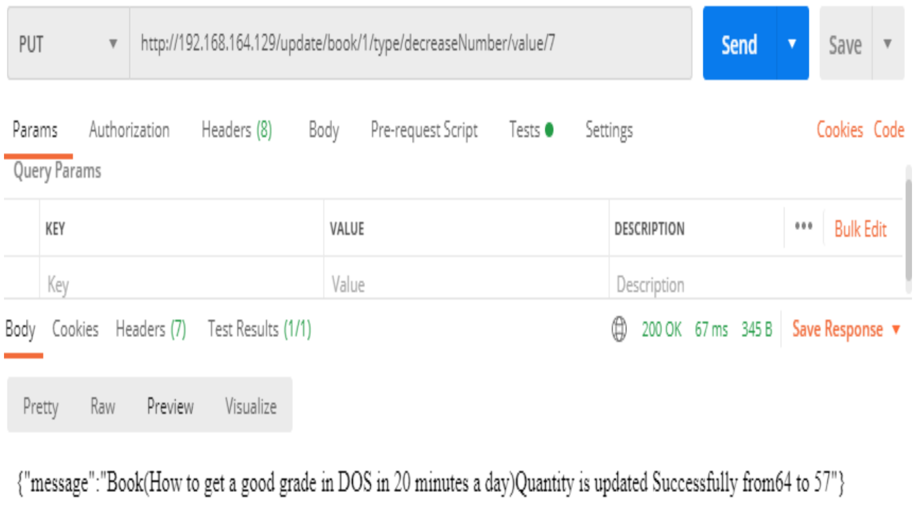
```
▼ 0:
    id:          "1"
    topic:       "distributed systems"
    title:       "How to get a good grade in DOS in 20 minutes a day"
    price:       "88.0"
    quantity:    "4"
```



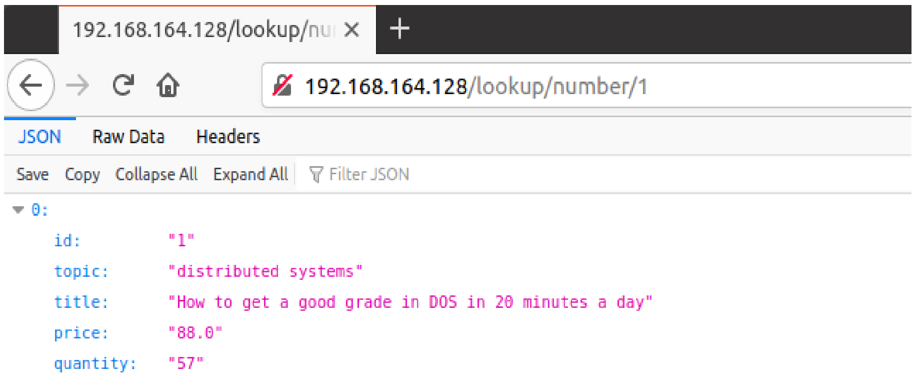
```
▼ 0:
    id:          "1"
    topic:       "distributed systems"
    title:       "How to get a good grade in DOS in 20 minutes a day"
    price:       "88.0"
    quantity:    "4"
```



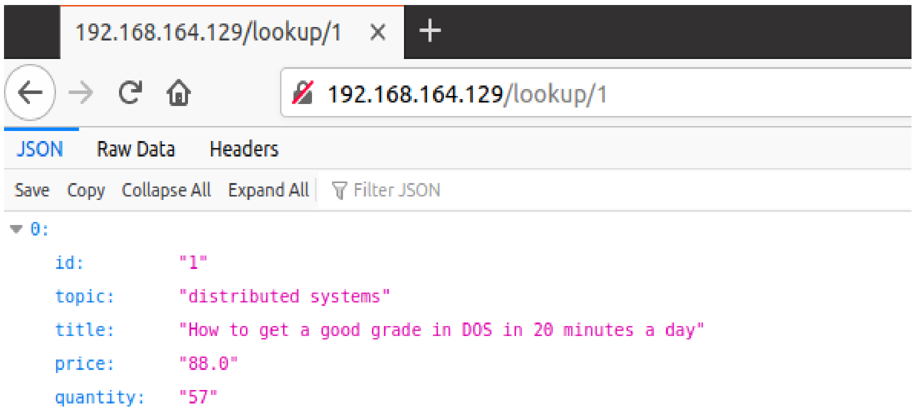
```
0:
    id:         "1"
    topic:      "distributed systems"
    title:      "How to get a good grade in DOS in 20 minutes a day"
    price:      "88.0"
    quantity:   "4"
```



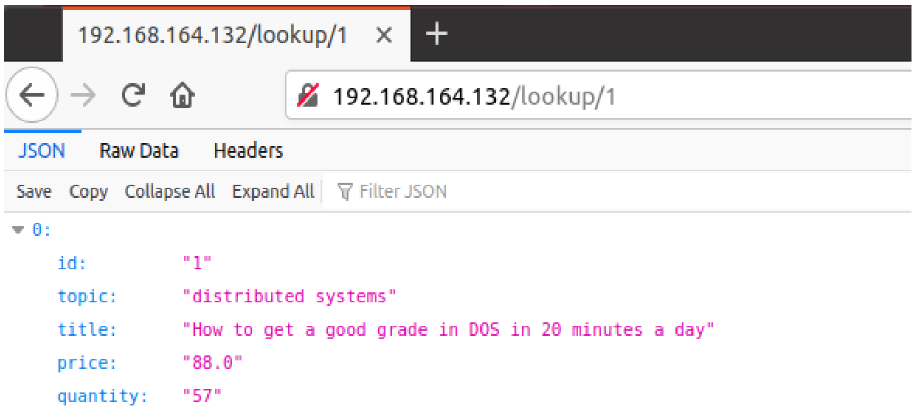
```
{"message":"Book(How to get a good grade in DOS in 20 minutes a day)Quantity is updated Successfully from64 to 57"}
```



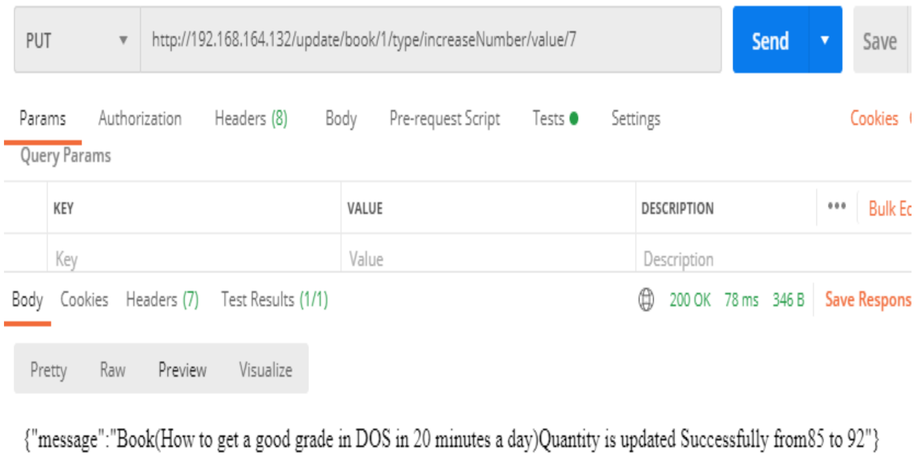
```
0:
    id:         "1"
    topic:      "distributed systems"
    title:      "How to get a good grade in DOS in 20 minutes a day"
    price:      "88.0"
    quantity:   "57"
```



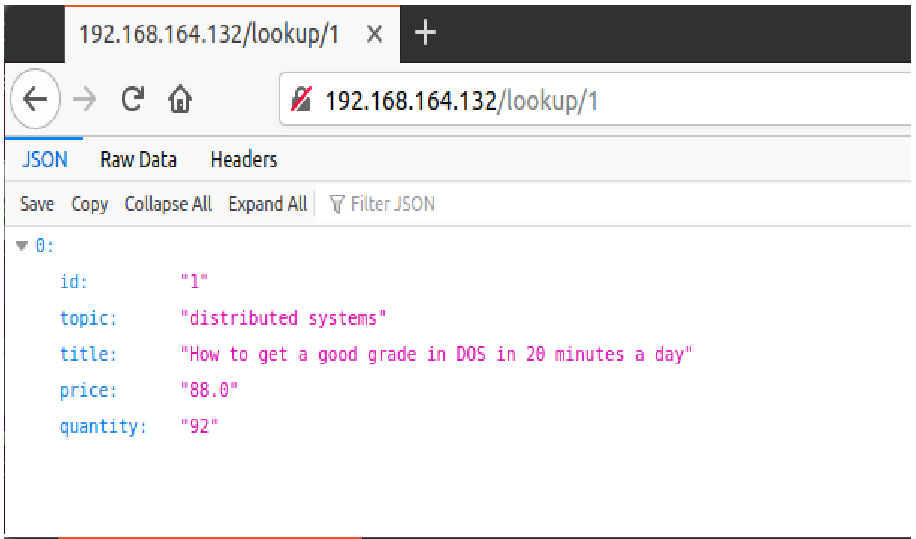
```
0:
    id:         "1"
    topic:      "distributed systems"
    title:      "How to get a good grade in DOS in 20 minutes a day"
    price:      "88.0"
    quantity:   "57"
```



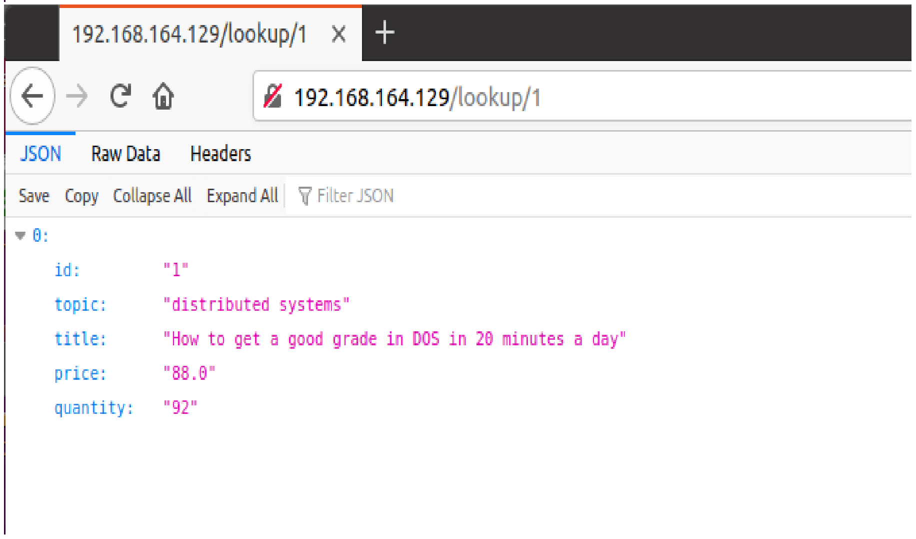
```
0:
    id:        "1"
    topic:     "distributed systems"
    title:     "How to get a good grade in DOS in 20 minutes a day"
    price:     "88.0"
    quantity:  "57"
```



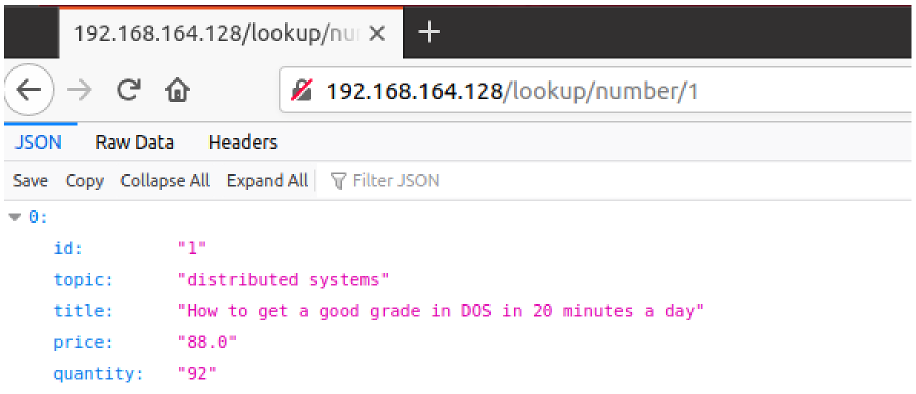
```
{"message":"Book(How to get a good grade in DOS in 20 minutes a day)Quantity is updated Successfully from85 to 92"}
```



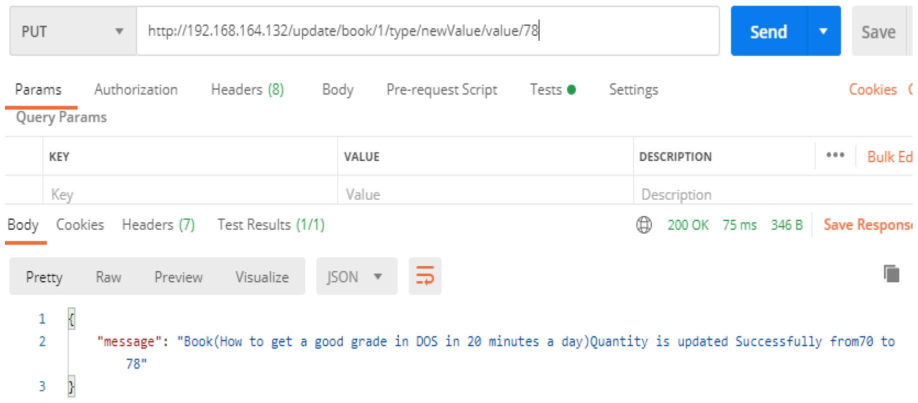
```
▼ 0:
    id:         "1"
    topic:      "distributed systems"
    title:      "How to get a good grade in DOS in 20 minutes a day"
    price:      "88.0"
    quantity:   "92"
```



```
▼ 0:
    id:         "1"
    topic:      "distributed systems"
    title:      "How to get a good grade in DOS in 20 minutes a day"
    price:      "88.0"
    quantity:   "92"
```

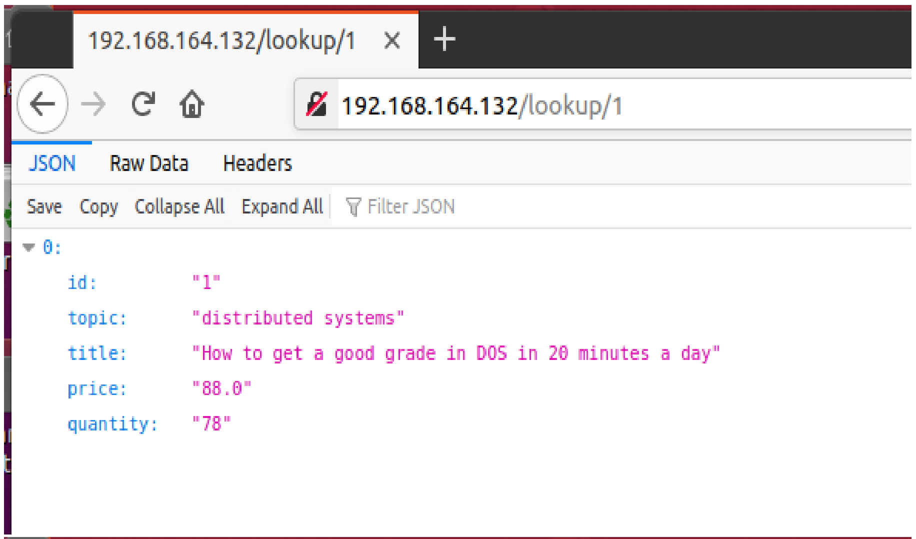


```
0:
    id:         "1"
    topic:      "distributed systems"
    title:      "How to get a good grade in DOS in 20 minutes a day"
    price:      "88.0"
    quantity:   "92"
```

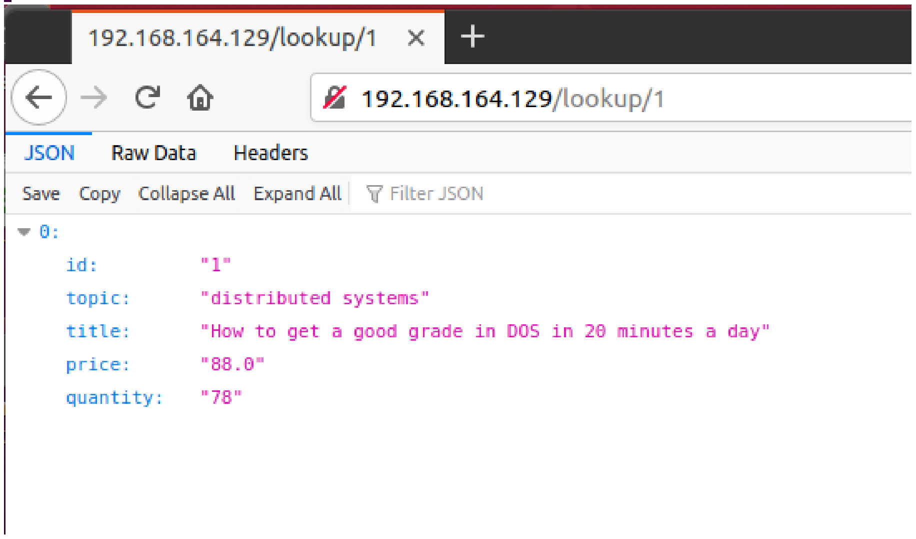


```
http://192.168.164.132/update/book/1/type/newValue/value/78
```

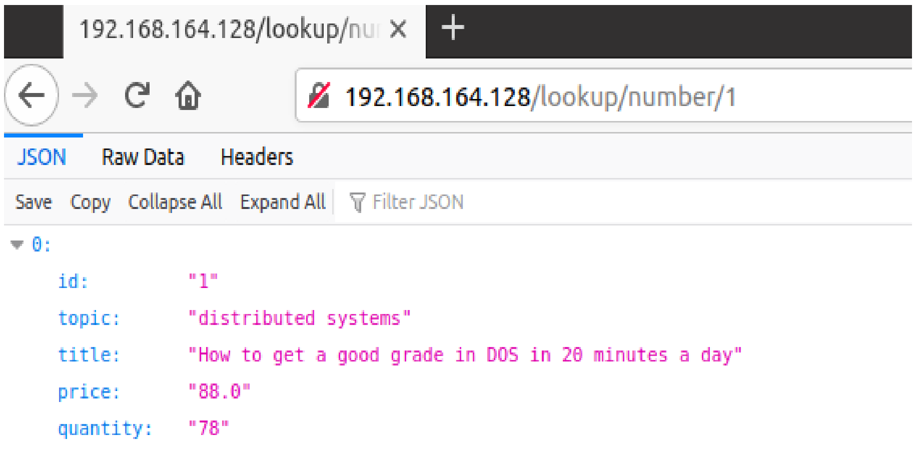
```
{
    "message": "Book(How to get a good grade in DOS in 20 minutes a day)Quantity is updated Successfully from70 to
      78"
}
```



```
▼ 0:
    id:         "1"
    topic:      "distributed systems"
    title:      "How to get a good grade in DOS in 20 minutes a day"
    price:      "88.0"
    quantity:   "78"
```

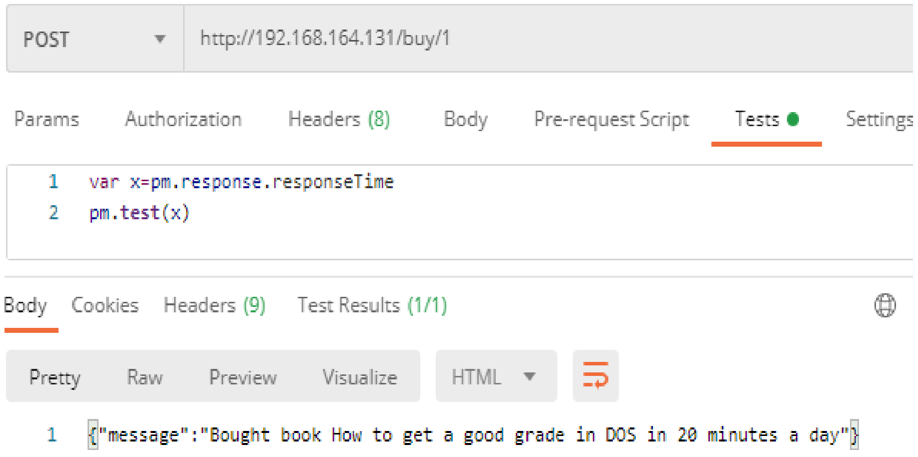


```
▼ 0:
    id:         "1"
    topic:      "distributed systems"
    title:      "How to get a good grade in DOS in 20 minutes a day"
    price:      "88.0"
    quantity:   "78"
```



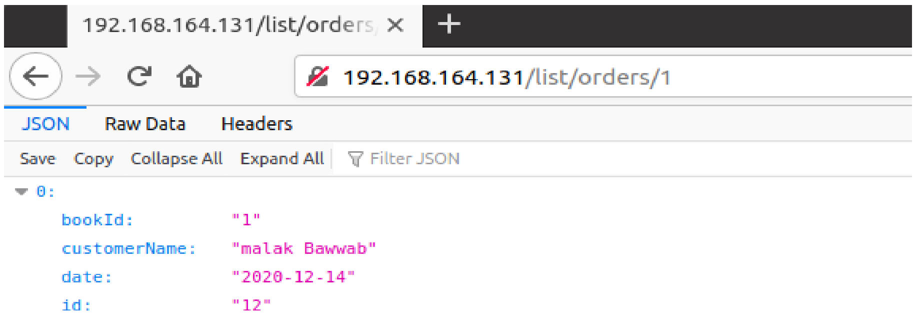
```
0:
    id:        "1"
    topic:     "distributed systems"
    title:     "How to get a good grade in DOS in 20 minutes a day"
    price:     "88.0"
    quantity:  "78"
```

Run the orders service ,replica:
*If the buy operation is successful,the same order record will be inserted in the order replicas.



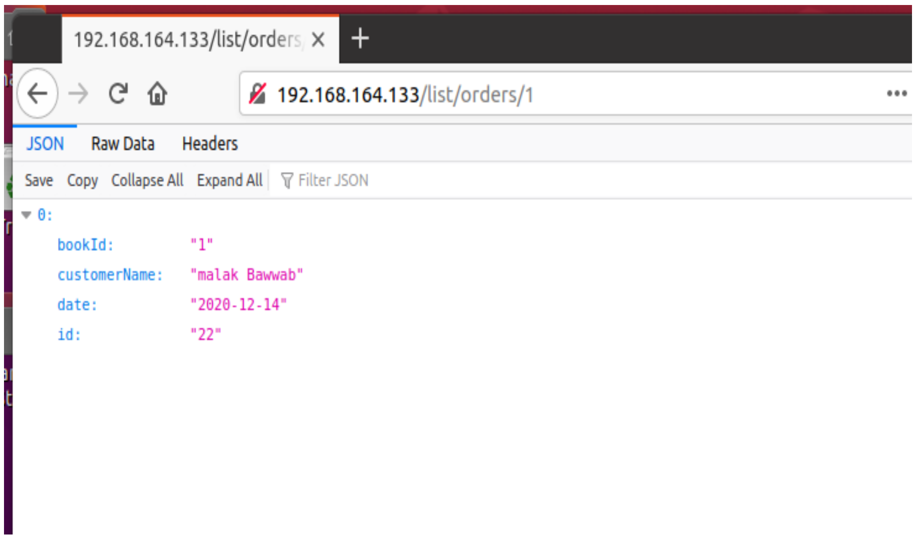
```
var x=pm.response.responseTime
pm.test(x)
```

```
{"message":"Bought book How to get a good grade in DOS in 20 minutes a day"}
```



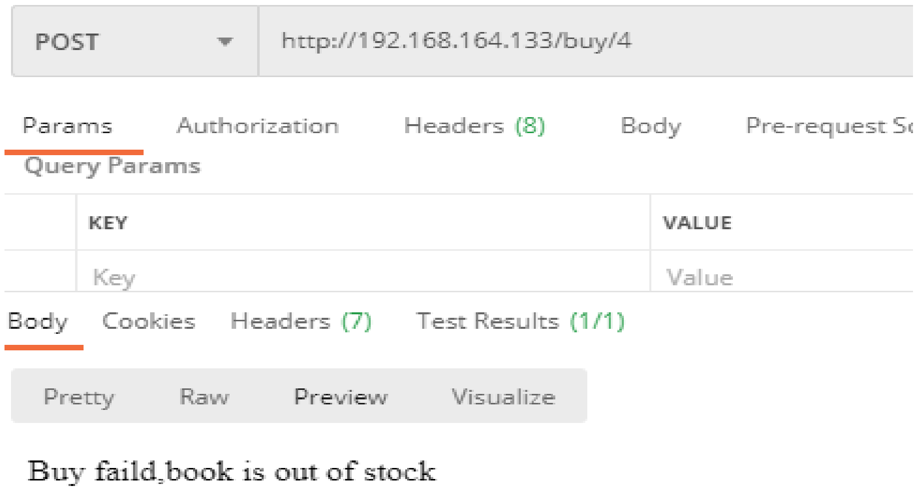
192.168.164.131/list/orders/1

```
0:
    bookId:         "1"
    customerName:   "malak Bawwab"
    date:           "2020-12-14"
    id:             "12"
```



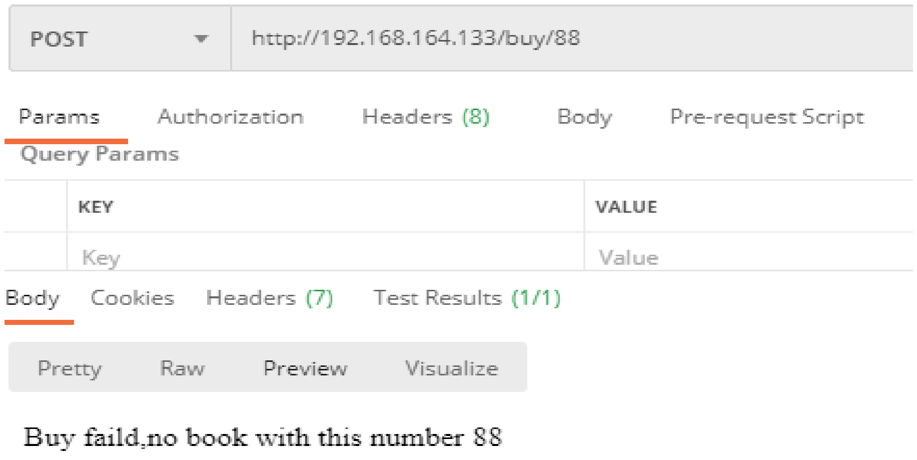
192.168.164.133/list/orders/1

```
0:
    bookId:         "1"
    customerName:   "malak Bawwab"
    date:           "2020-12-14"
    id:             "22"
```



http://192.168.164.133/buy/4

Buy faild,book is out of stock



http://192.168.164.133/buy/88

Buy faild,no book with this number 88

7 Flow

- When a search request occur,the front end server check if the cache has the topic or not,if yes it will return the result.If not,front end will send a request to one of the catalog replicas based on round-robin algorithm to fetch the data from the DB ,and it will store the returned data in the cache.
- When a lookup request occur,the front end server check if the cache has the itemNumber or not,if yes it will return the result.If not,front end will send a request to one of the catalog

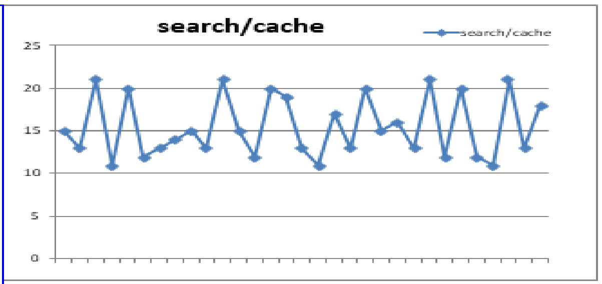
replicas based on round-robin algorithm to fetch the data from the DB ,and it will store the returned data in the cache.

- Successful Buy:when buy operation occur,the front end will send a request to one of the order replicas based on the round-robin algorithms,order server will send a request to check if the cache contain itemNumber or not(if not,check the DB also) to know if the book exists on the store or not,Then if the book exists,order server will send update request to catalog to decrease quantity by one,also order server will insert an order record on orders table and will send a notify request to other order replicas to insert the same order record on their DBs.
- Updating quantity/cost in catalog will send an invalidate request to remove item from cache and will send a notify update request to other catalog replicas with the new updated value.

8 Performance Results

- Cache enhance response time effectively (more than twice).for both search/lookup request.
- But in buy request,cache doesn't enhance performance very much ,because in buy request (there is an update to the quantity,when the buy operation is successful,the quantity of the book is decreased by 1)and in case of updates ,an invalidate request will be sent to remove item from the cache.
- So cache work well and enhance performance effectively in queries operations not in update operations (writes to databases).Because updates causes the item to be removed from the cache according to server push technique with invalidate approach.
- I use postman to git accurate response time.

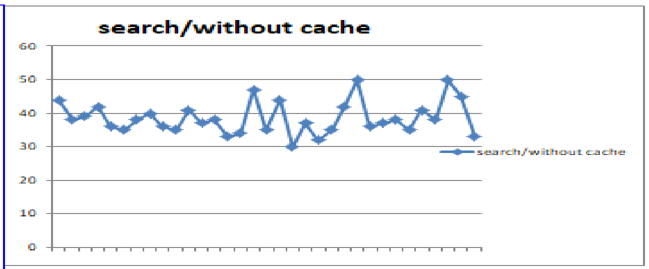
search/cache
15
13
21
11
20
12
13
14
15
13
21
15
12
20
19
13
11
17
13
20
15
16
13
21
12
20
12
11
21
13
18



AV.ResponseTime=15.48387

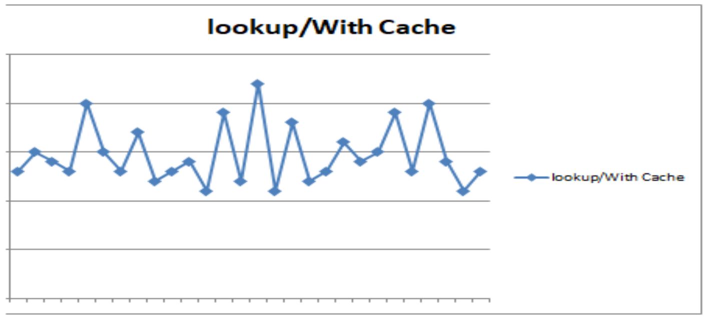
1

search/without cache
44
38
39
42
36
35
38
40
36
35
41
37
38
33
34
47
35
44
30
37
32
35
42
50
36
37
38
35
41
38
50
45
33



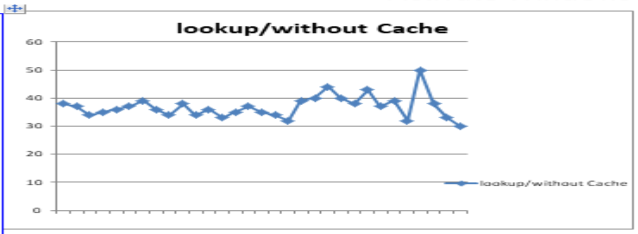
AV.ResponseTime=38.51515ms

14.714286
lookup/With Cache(ms)
13
15
14
13
20
15
13
17
12
13
14
11
19
12
22
11
18
12
13
16
14
15
19
13
20
14
11
13

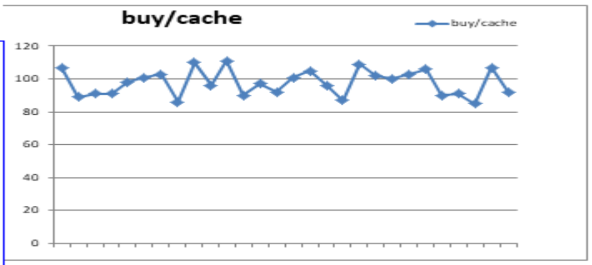


AV.Response Time is 14.714286 ms

lookup/without Cache(ms)
38
37
34
35
36
37
39
36
34
38
34
36
33
35
37
35
34
32
39
40
44
40
38
43
37
39
32
50
38
33
30

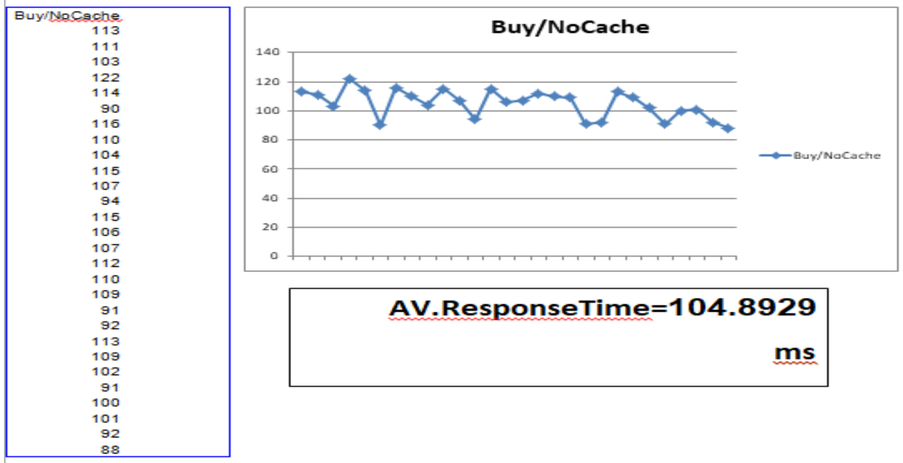


AV.Response time=36.87096 ms



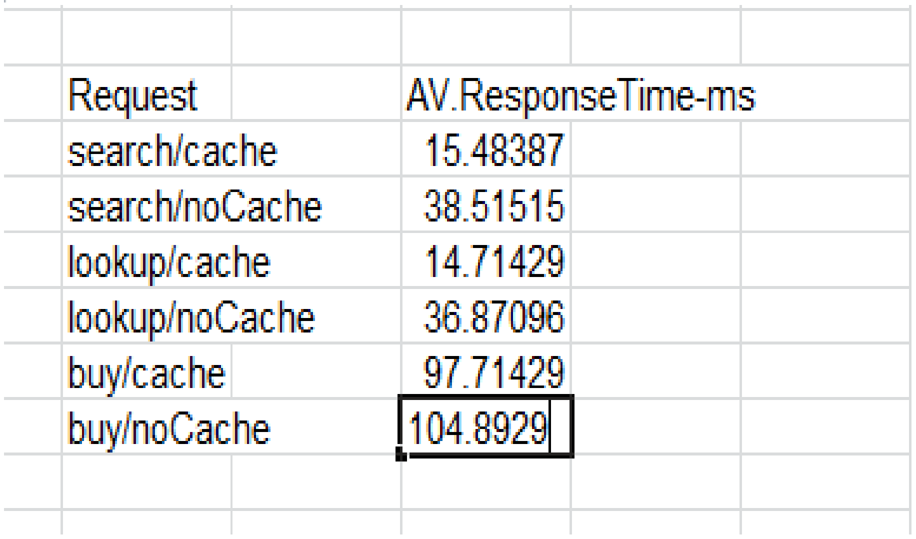
buy/cache
107
89
91
91
98
101
103
86
110
96
111
90
97
92
101
105
96
87
109
102
100
103
106
90
91
85
107
92

AV.Response Time=97.714286



Buy/NoCache
113
111
103
122
114
90
116
110
104
115
107
94
115
106
107
112
110
109
91
92
113
109
102
91
100
101
92
88

AV.ResponseTime=104.8929 ms



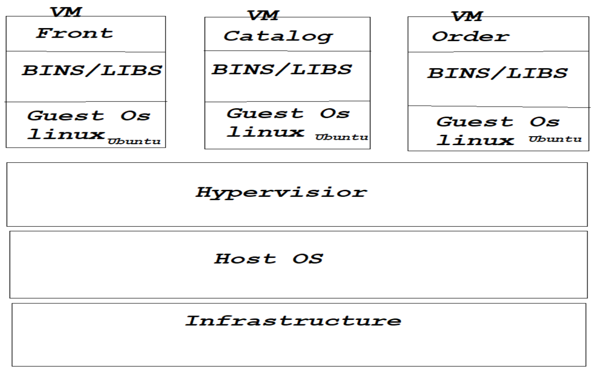
Request	AV.ResponseTime-ms
search/cache	15.48387
search/noCache	38.51515
lookup/cache	14.71429
lookup/noCache	36.87096
buy/cache	97.71429
buy/noCache	104.8929

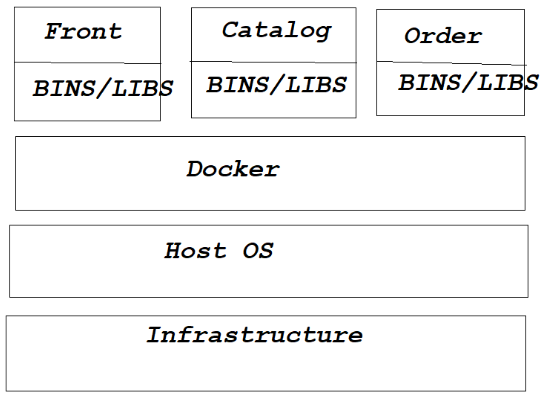
9 Improvements and design tradoffs

- Implement another approach instead of a push-based approach with invalidate.In push based-approach ,there is a burden on the server,and there is a need to maintain state at the server (since HTTP is stateless ,we need mechanisms beyond HTTP).
- Implement another load balancing algorithm rather than round robin(round robin distribute requests evenly between the servers).
- Add some caching features such as a limit on the number of items in the cache, which will then need a cache replacement

policy such as LRU to replace older items with newer ones.

- Use Docker instead of 3 instances of VMs with ubuntu because:
  - Docker containers are lightweight bit VMs are heavyweight.
  - Virtual machines have host OS and the guest OS inside each VM. In contrast, Docker containers host on a single physical server with a host OS, which shares among them. Sharing the host OS between containers makes them light .
  - VMs are slow, each VM includes a full copy of an operating system, the application, and necessary binaries and libraries — taking up tens of GBs.
  - Containers take up less space than VMs .
  - Docker eliminate the need for a guest OS and we save the amount of memory that was occupied by it.


VM
Front
BINS/LIBS
Guest Os
linux Ubuntu
VM
Catalog
BINS/LIBS
Guest Os
linux Ubuntu
VM
Order
BINS/LIBS
Guest Os
linux Ubuntu
Hypervisior
Host OS
Infrastructure


Front
BINS/LIBS
Catalog
BINS/LIBS
Order
BINS/LIBS
Docker
Host OS
Infrastructure