
Project Report



AL-NAJAH UNIVERSITY

DOS

SUBMITTED TO:

Dr.Samer Arandi

PREPARED BY:

Shahd Khader

12027906

Ahmad Nazzal

12027687

Introduciton:

In this lab, I restructured the online bookstore, Bazar.com, which I developed in Lab 1. The goal is to improve request processing time as the system struggles to handle a higher volume of requests due to increased customer demand. To achieve this, I introduced replication, caching, and other design improvements to create a more scalable and efficient architecture. This project also aims to teach concepts related to multi-tier web design, microservices, and containerization using Docker.

The following report details the modifications made to the system to support replication, caching, and consistency. Additionally, I describe the technical solutions implemented to reduce latency and enhance scalability.

New Architecture with Replication and Caching

To improve performance, I implemented a multi-tier architecture with the following components:

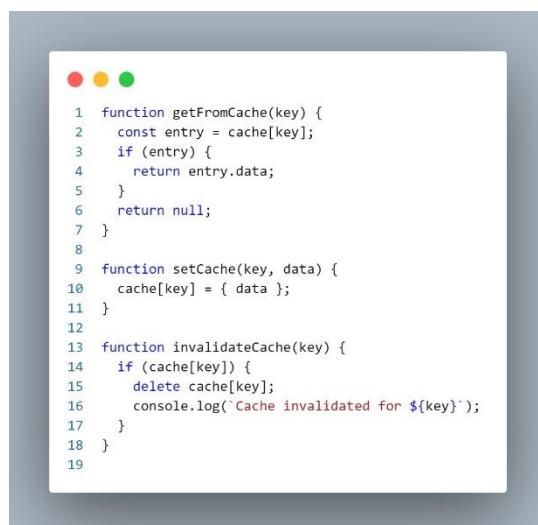
- **Front-End Server with In-Memory Cache:** The front-end now includes an in-memory cache to store frequently accessed catalog data and recent orders.
- **Replicated Catalog and Order Services:** I added multiple replicas of the catalog and order services to improve redundancy and decrease response times by distributing the load.

Implementation Details

In-Memory Caching

To reduce response time, an in-memory cache was implemented on the front-end server. This cache stores recent requests to avoid repeated database queries for the same data.

- **Cache Design:** The cache is implemented as a key-value store. Frequently requested items (e.g., popular books) and recent orders are stored in memory.



```
1 function getFromCache(key) {
2   const entry = cache[key];
3   if (entry) {
4     return entry.data;
5   }
6   return null;
7 }
8
9 function setCache(key, data) {
10   cache[key] = { data };
11 }
12
13 function invalidateCache(key) {
14   if (cache[key]) {
15     delete cache[key];
16     console.log(`Cache invalidated for ${key}`);
17   }
18 }
19
```

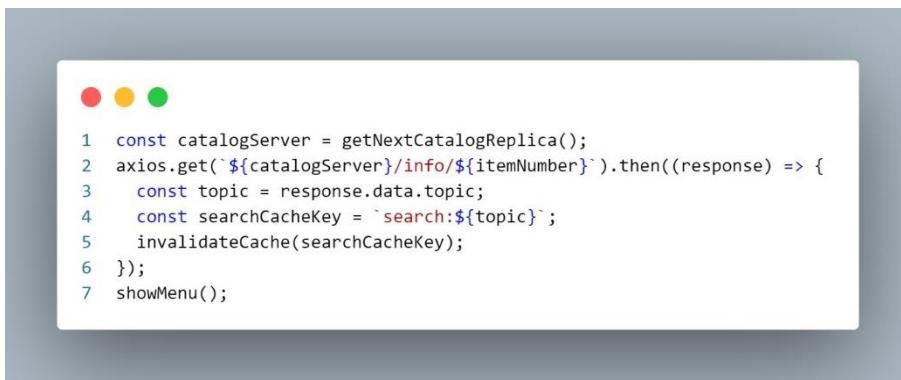
Data Replication

Data replication was added to improve availability and reduce latency. Multiple instances of the catalog and order services were created, with the front-end server distributing requests across these instances.

- **Replication Strategy:** I used a round-robin method for distributing requests to replicas.
- **Load Balancing:** The front-end server sends requests to catalog and order service replicas to balance the workload.



```
● ● ●
1 const cache = {};
2
3 const catalogReplicas = ["http://catalog-service-1:3001", "http://catalog-service-2:3002"];
4 const orderReplicas = ["http://order-service-1:3003", "http://order-service-2:3004"];
5
6 let catalogReplicaIndex = 0;
7 let orderReplicaIndex = 0;
8
9 function getNextCatalogReplica() {
10   catalogReplicaIndex = (catalogReplicaIndex + 1) % catalogReplicas.length;
11   console.log(catalogReplicas[catalogReplicaIndex]);
12   return catalogReplicas[catalogReplicaIndex];
13 }
14
15 function getNextOrderReplica() {
16   orderReplicaIndex = (orderReplicaIndex + 1) % orderReplicas.length;
17   console.log(orderReplicas[orderReplicaIndex]);
18   return orderReplicas[orderReplicaIndex];
19 }
```



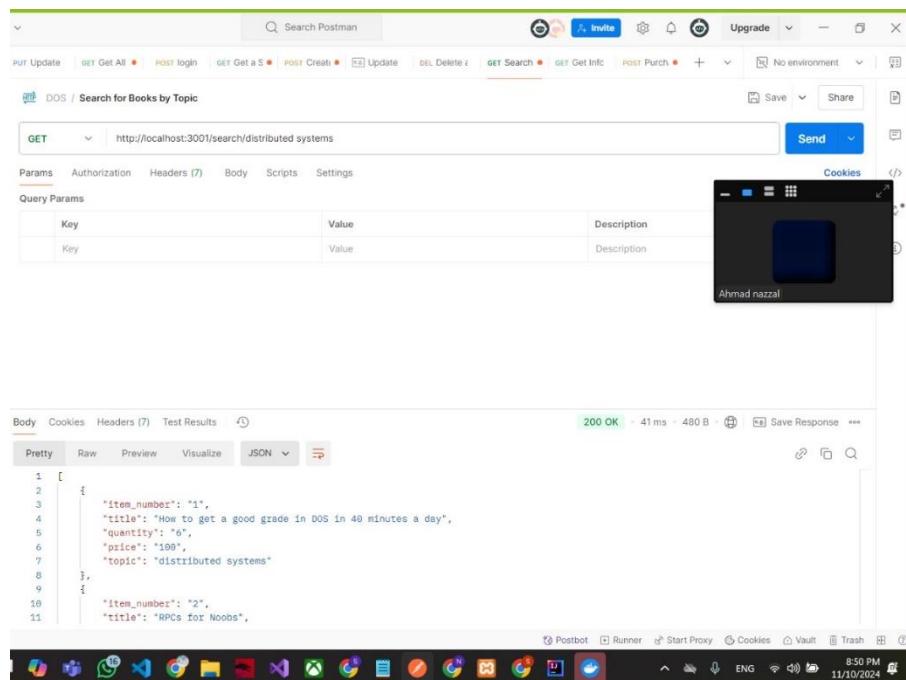
```
● ● ●
1 const catalogServer = getNextCatalogReplica();
2 axios.get(`${catalogServer}/info/${itemNumber}`).then((response) => {
3   const topic = response.data.topic;
4   const searchCacheKey = `search:${topic}`;
5   invalidateCache(searchCacheKey);
6 });
7 showMenu();
```

Testing and Results

Latency Improvement

I measured the response times before and after implementing caching and replication. After the changes, the average response time decreased significantly, as shown in the following results:

Without using cache



DOS / Search for Books by Topic

GET http://localhost:3001/search/distributed systems

Params Authorization Headers (7) Body Scripts Settings

Query Params

Key	Value	Description
Key	Value	Description

Cookies

Ahmad nazzal

Body Cookies Headers (7) Test Results

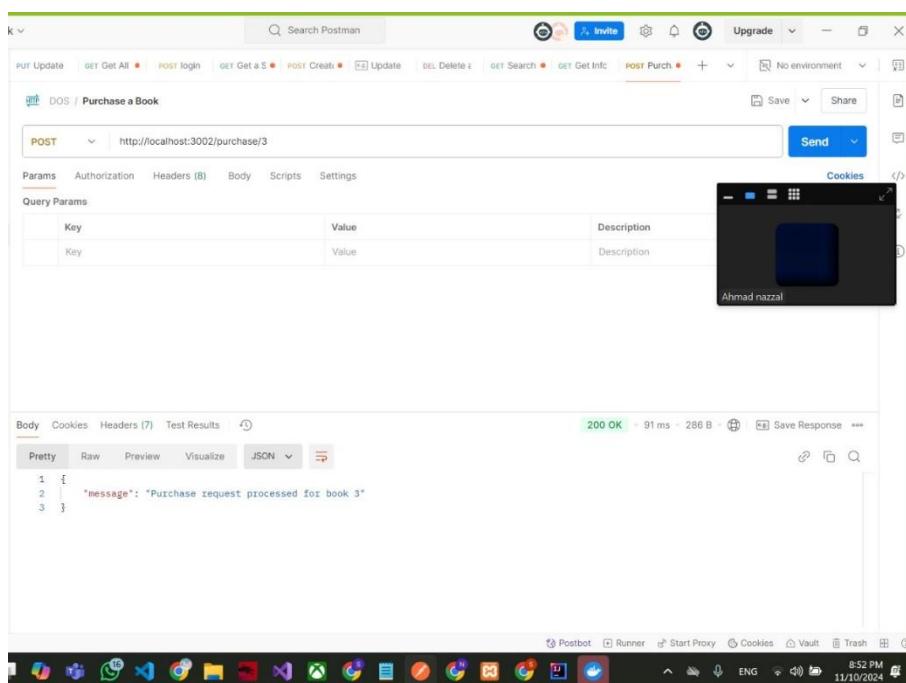
Pretty Raw Preview Visualize JSON

```
[{"item_number": "1", "title": "How to get a good grade in DOS in 40 minutes a day", "quantity": "6", "price": "100", "topic": "distributed systems"}, {"item_number": "2", "title": "RPCs for Noobs", "quantity": "1", "price": "50", "topic": "distributed systems"}]
```

200 OK 41 ms 480 B Save Response

Postbot Runner Start Proxy Cookies Vault Trash

8:50 PM 11/10/2024



DOS / Purchase a Book

POST http://localhost:3002/purchase/3

Params Authorization Headers (8) Body Scripts Settings

Query Params

Key	Value	Description
Key	Value	Description

Cookies

Ahmad nazzal

Body Cookies Headers (7) Test Results

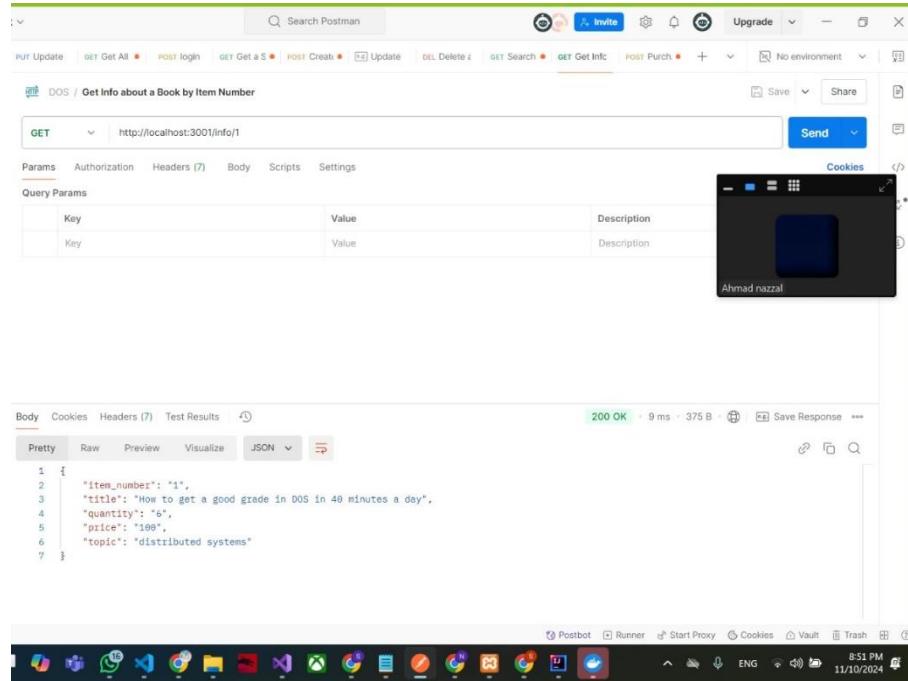
Pretty Raw Preview Visualize JSON

```
{"message": "Purchase request processed for book 3"}
```

200 OK 91 ms 286 B Save Response

Postbot Runner Start Proxy Cookies Vault Trash

8:52 PM 11/10/2024



DOS / Get Info about a Book by Item Number

GET http://localhost:3001/info/1

Params Authorization Headers (7) Body Scripts Settings

Query Params

Key	Value	Description
Key	Value	Description

Cookies

Ahmad nazzal

Body Cookies Headers (7) Test Results

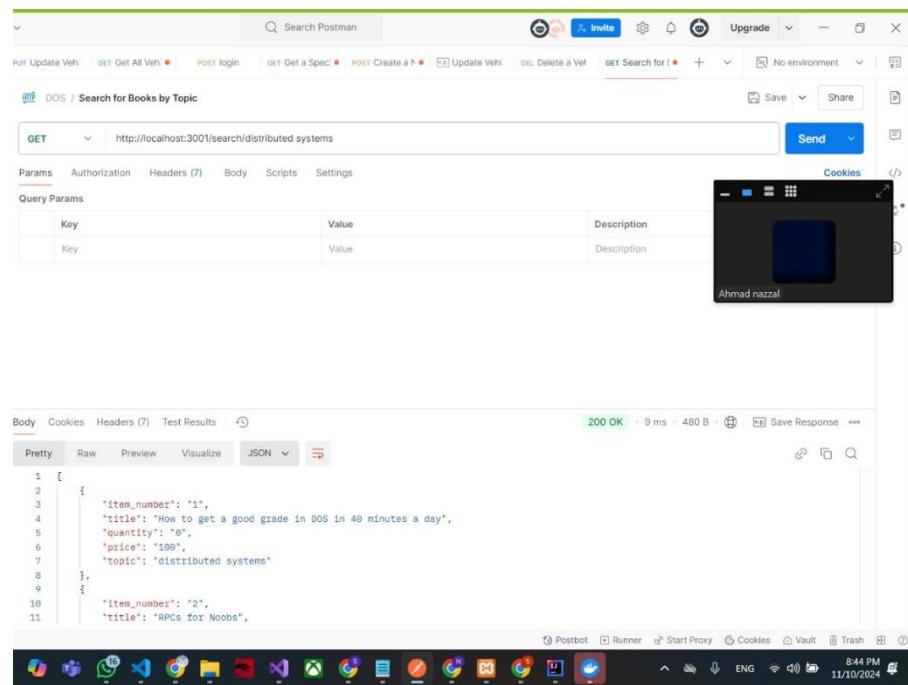
Pretty Raw Preview Visualize JSON

```
1 {
2   "item_number": "1",
3   "title": "How to get a good grade in DOS in 40 minutes a day",
4   "quantity": "6",
5   "price": "100",
6   "topic": "distributed systems"
7 }
```

200 OK 9 ms 375 B Save Response

Postbot Runner Start Proxy Cookies Vault Trash 8:51 PM 11/10/2024

With using cache



DOS / Search for Books by Topic

GET http://localhost:3001/search/distributed systems

Params Authorization Headers (7) Body Scripts Settings

Query Params

Key	Value	Description
Key	Value	Description

Cookies

Ahmad nazzal

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "item_number": "1",
4     "title": "How to get a good grade in DOS in 40 minutes a day",
5     "quantity": "6",
6     "price": "100",
7     "topic": "distributed systems"
8   },
9   {
10     "item_number": "2",
11     "title": "RPCs for Noobs",
12   }
]
```

200 OK 9 ms 480 B Save Response

Postbot Runner Start Proxy Cookies Vault Trash 8:44 PM 11/10/2024

Body

```

1 [
2   "item_number": "1",
3   "title": "How to get a good grade in DOS in 40 minutes a day",
4   "quantity": "8",
5   "price": "100",
6   "topic": "distributed systems"
7 ]

```

Body

```

1 [
2   "message": "Purchase request processed for book 3"
3 ]

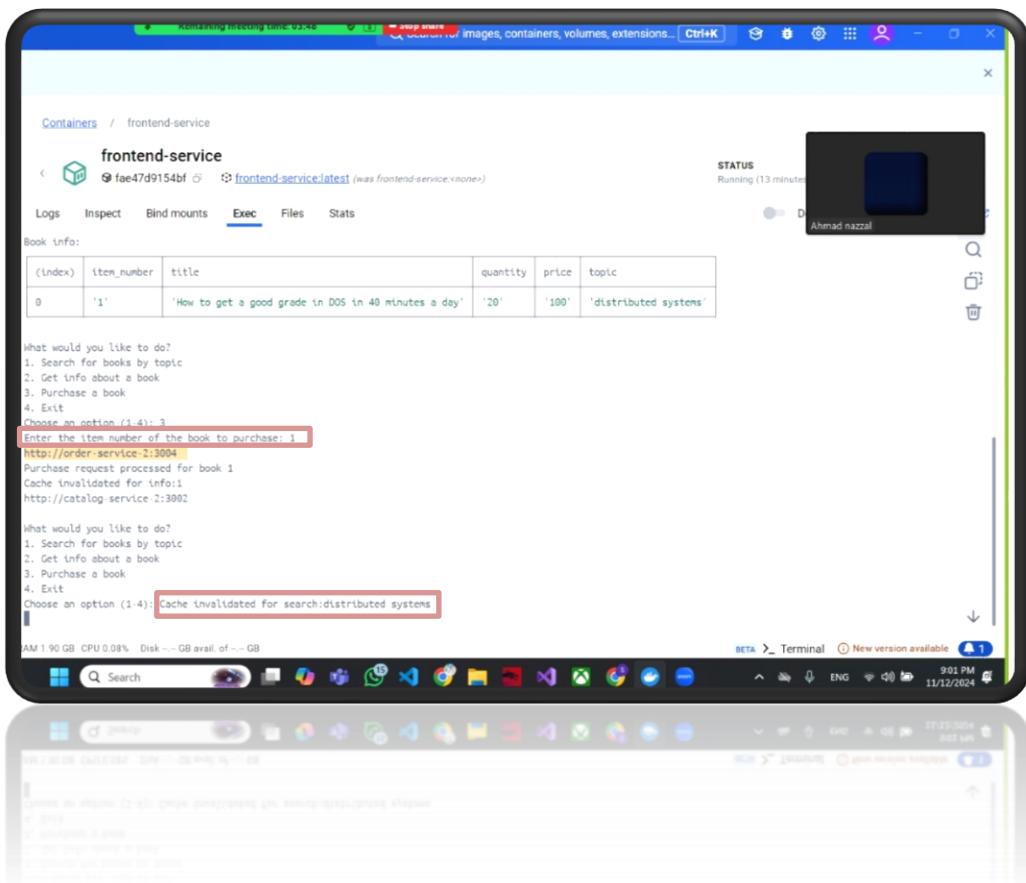
```

The table that shows the differences

Feature	Before (ms)	Before (ms)
Search by topic – Catalog	41	9
Search by id - Catalog	9	6
Purchase - Order	91	14

Consistency Testing

Consistency was verified by simulating concurrent requests. For example, if a book's stock was updated in one replica, all replicas reflected this change within a short time to maintain data integrity.



Optional: Containerization with Docker

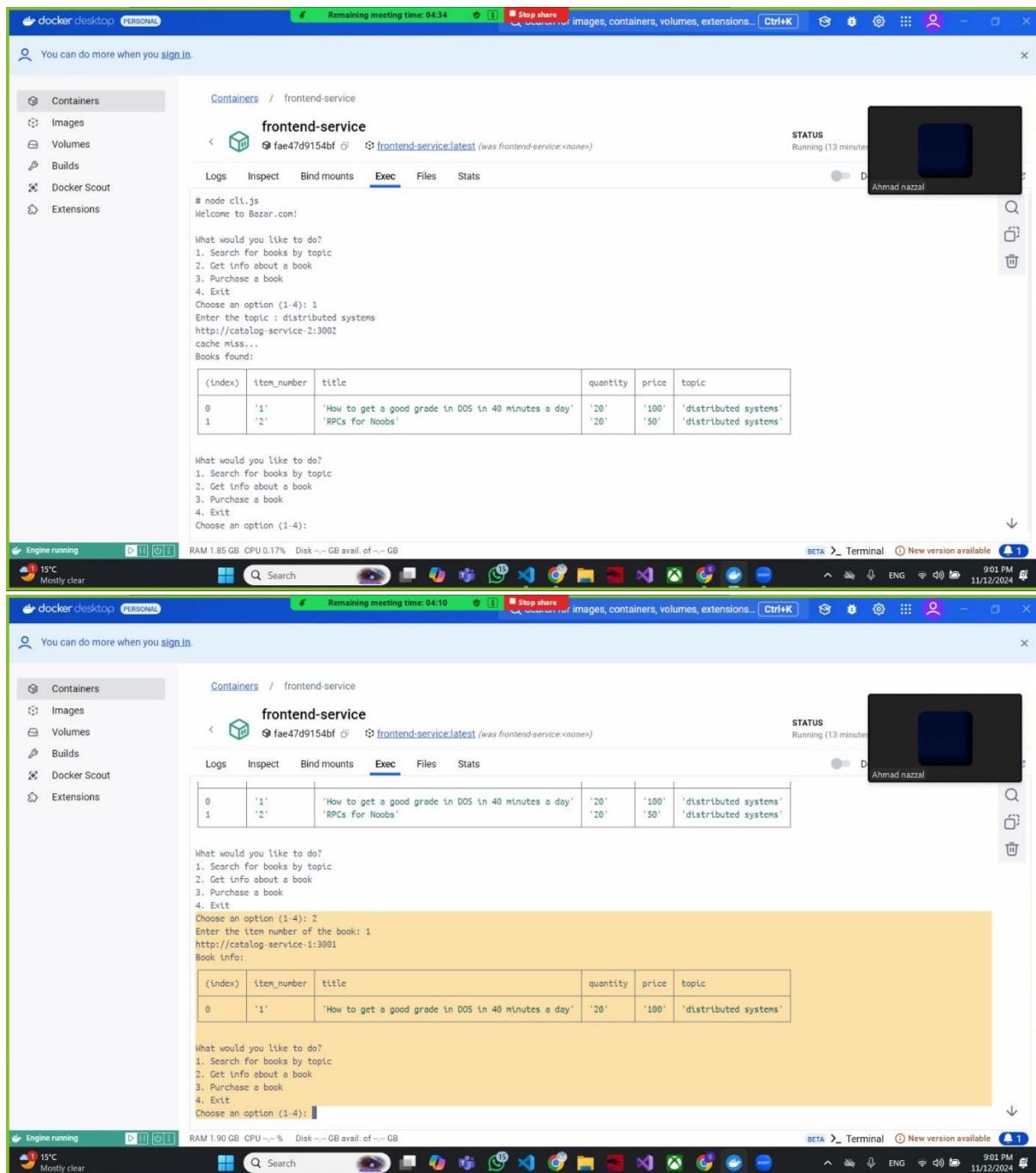
To make the system easily deployable, I containerized each service using Docker. This setup enables the application to run in isolated containers, making it easy to deploy and scale.

Docker-Compose File:

```
1 version: "3"
2 services:
3   catalog-service-1:
4     image: catalog-service
5     container_name: catalog-service-1
6     build:
7       context: ./catalog-service
8     ports:
9       - "3001:3001"
10    environment:
11      - PORT=3001
12    volumes:
13      - ./catalog-service/catalog.csv:/app/catalog.csv
14    networks:
15      - network1
16
17   catalog-service-2:
18     image: catalog-service
19     container_name: catalog-service-2
20     ports:
21       - "3002:3001"
22     environment:
23       - PORT=3002
24     volumes:
25       - ./catalog-service/catalog.csv:/app/catalog.csv
26     networks:
27       - network1
28
29   order-service-1:
30     image: order-service
31     container_name: order-service-1
32     build:
33       context: ./order-service
34     ports:
35       - "3003:3003"
36     environment:
37       - PORT=3003
38     networks:
39       - network1
40     depends_on:
41       - catalog-service-1
42       - catalog-service-2
43
44   order-service-2:
45     image: order-service
46     container_name: order-service-2
47     ports:
48       - "3004:3003"
49     environment:
50       - PORT=3004
51     networks:
52       - network1
53     depends_on:
54       - catalog-service-1
55       - catalog-service-2
56
57   frontend-service:
58     image: frontend-service
59     container_name: frontend-service
60     build:
61       context: ./frontend
62     networks:
63       - network1
64     depends_on:
65       - order-service-1
66       - order-service-2
67       - catalog-service-1
68       - catalog-service-2
69     stdin_open: true
70     tty: true
71
72   networks:
73     network1:
74       driver: bridge
75
```

Result

The images below show each request sent to different replica and save the consistency between the containers



You can do more when you sign in.

Containers / frontend-service

frontend-service

fae47d9154bf ⌂ fae47d9154bf (was frontend-service<none>)

Logs Inspect Bind mounts Exec Files Stats

STATUS
Running (13 minutes)
Ahmad nazzal

4. Exit
Choose an option (1-4): 3
Enter the item number of the book to purchase: 1
<http://order-service:2:3004>
Purchase request processed for book 1
Cache invalidated for info:1
<http://catalog-service-2:3002>

What would you like to do?
1. Search for books by topic
2. Get info about a book
3. Purchase a book
4. Exit
Choose an option (1-4): 3
Enter the item number of the book to purchase: 1
<http://order-service:1:3003>
Purchase request processed for book 1
<http://catalog-service-1:3001>

What would you like to do?
1. Search for books by topic
2. Get info about a book
3. Purchase a book
4. Exit
Choose an option (1-4):

Book info:

(index)	item_number	title	quantity	price	topic
0	'1'	'How to get a good grade in DOS in 40 minutes a day'	'20'	'100'	'distributed systems'

What would you like to do?
1. Search for books by topic
2. Get info about a book
3. Purchase a book
4. Exit
Choose an option (1-4): 3
Enter the item number of the book to purchase: 1
<http://order-service:2:3004>
Purchase request processed for book 1
Cache invalidated for info:1
<http://catalog-service-2:3002>

What would you like to do?
1. Search for books by topic
2. Get info about a book
3. Purchase a book
4. Exit
Choose an option (1-4): Cache invalidated for search:distributed systems

Engine running RAM 1.94 GB CPU 0.33% Disk --- GB avail. of --- GB

15°C Mostly clear

Search Terminal New version available 1 9:02 PM 11/12/2024

You can do more when you sign in.

Containers / frontend-service

frontend-service

fae47d9154bf ⌂ fae47d9154bf (was frontend-service<none>)

Logs Inspect Bind mounts Exec Files Stats

STATUS
Running (13 minutes)
Ahmad nazzal

4. Exit
Choose an option (1-4): 3
Enter the item number of the book to purchase: 1
<http://order-service:2:3004>
Purchase request processed for book 1
Cache invalidated for info:1
<http://catalog-service-2:3002>

What would you like to do?
1. Search for books by topic
2. Get info about a book
3. Purchase a book
4. Exit
Choose an option (1-4): 3
Enter the item number of the book to purchase: 1
<http://order-service:1:3003>
Purchase request processed for book 1
<http://catalog-service-1:3001>

What would you like to do?
1. Search for books by topic
2. Get info about a book
3. Purchase a book
4. Exit
Choose an option (1-4):

Book info:

(index)	item_number	title	quantity	price	topic
0	'1'	'How to get a good grade in DOS in 40 minutes a day'	'20'	'100'	'distributed systems'
1	'2'	'How to get a good grade in C in 40 minutes a day'	'30'	'150'	'distributed systems'

What would you like to do?
1. Search for books by topic
2. Get info about a book
3. Purchase a book
4. Exit
Choose an option (1-4): 3
Enter the item number of the book to purchase: 1
<http://order-service:2:3004>
Purchase request processed for book 1
Cache invalidated for info:1
<http://catalog-service-2:3002>

What would you like to do?
1. Search for books by topic
2. Get info about a book
3. Purchase a book
4. Exit
Choose an option (1-4): Cache invalidated for search:distributed systems

Engine running RAM 1.90 GB CPU 0.08% Disk --- GB avail. of --- GB

15°C Mostly clear

Search Terminal New version available 1 9:01 PM 11/12/2024

Good Buy 😊