DOS - Project Part -1-

Distributed Microservices Application.

Distributed Operating Systems-10636456.

Haya Mikkawi
11819675
Tuesday, 29 March 2022

Overall Program Design:

In this part of the project, we were asked to implement three servers; a **catalog server** which directly communicates with the database (to get a book by number, get books by topic, and modify the price or the quantity of a specific book identified by its number).
The second server is the **order server**, which handles the purchase operations, but first it needs to communicate with the catalog server to check whether this book is available, it also needs to communicate with the catalog server to modify the quantity of the purchased book.
And finally, we have the **frontend server;** that's the server that the client directly communicates with, it only has to redirect these requests either to the catalog server or to the order server.
Each one of these servers was running on a different virtual machine as separate service, so I had 2 virtual machines in addition to my machine,
In my machine, I had the frontend server and the client which is Postman, in the first VM, I had the catalog server and the database deployed, and in the second VM, I had the order server.

I used MongoDB to store the books, for each book there is a record that includes: itemNumber, name, cost, topic and numberOfItems.

Used technologies: Node.js & express, mongoDB, NGinX, Parallels (for the VMs) and each VM had Ubuntu running on it.

How it works:

These are the exact steps of how the system works:

1- A request is sent by the client which can be the browser, Postman, etc. The client only talks to the frontend server (deployed on my machine) and has full transparency of what happens behind the scenes.
So, the clients will send requests to localhost on port 3002.

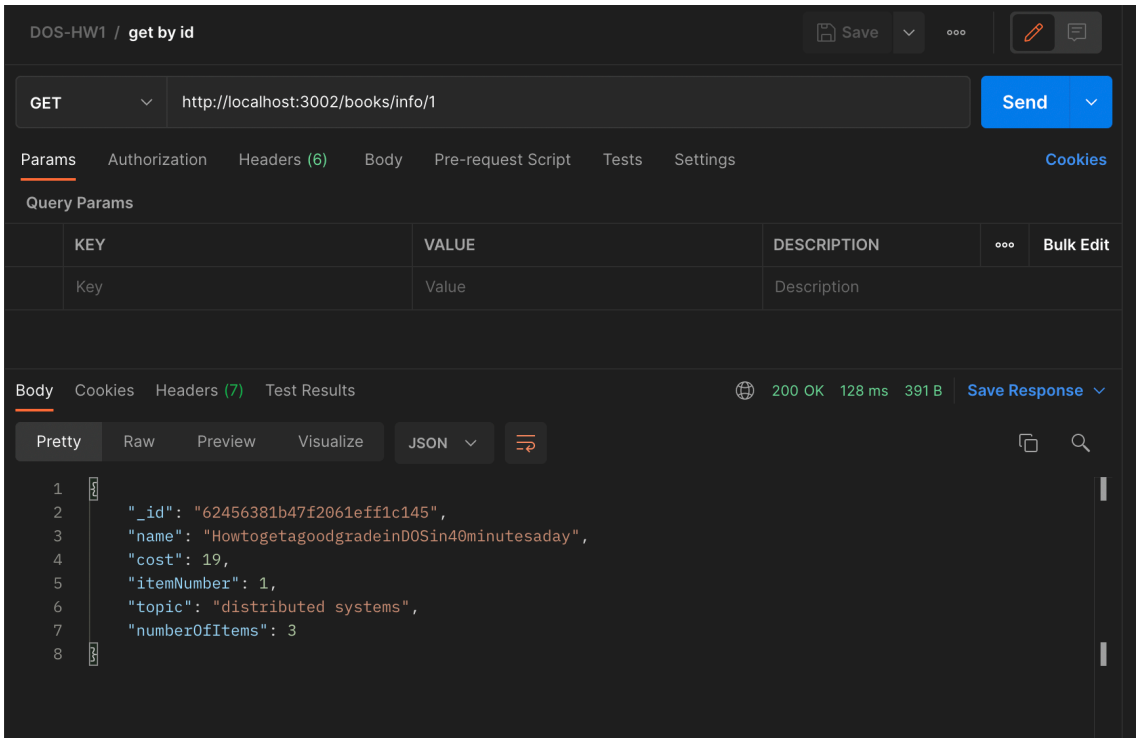
2- When the frontend server receives a request it redirects this request to one of the two servers depending on the request type; catalog server and order server, each one of these servers is running on a different VM.

3- Catalog server is the only server that is allowed to read/modify on the database and it handles three types of requests: get a book by its id, get books by their topic and modify the cost or number of items for a book, catalog server is running on a VM with IP address: 10.211.55.3, requests are sent to NginX on port 80 and redirected to port 3000.

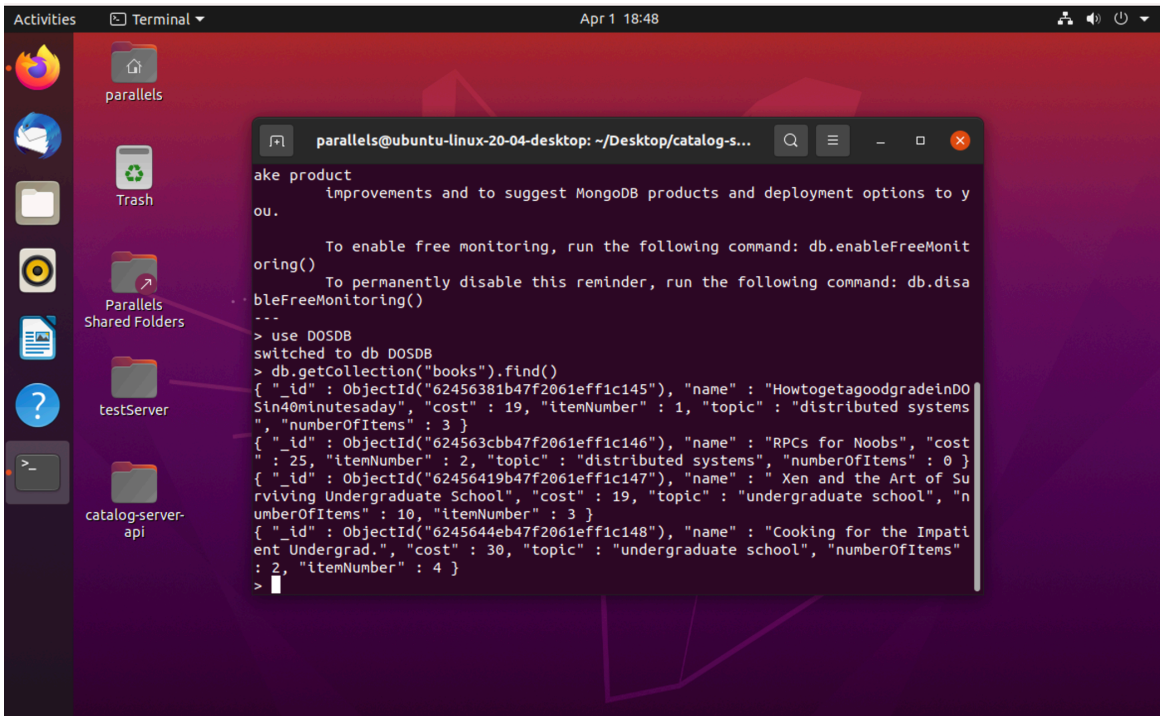
4- Order server only handles the purchase requests, it first communicates with the catalog server to check whether this item is available, if not then the operation isn't valid, otherwise, the order server needs to contact the catalog server again to decrement the quantity of that item.
catalog server is running on a VM with IP address: 10.211.55.4, requests are sent to NginX on port 80 and redirected to port 3000.

In each one of these servers NginX is running as a proxy server to redirect requests from port 80 to port 3000.

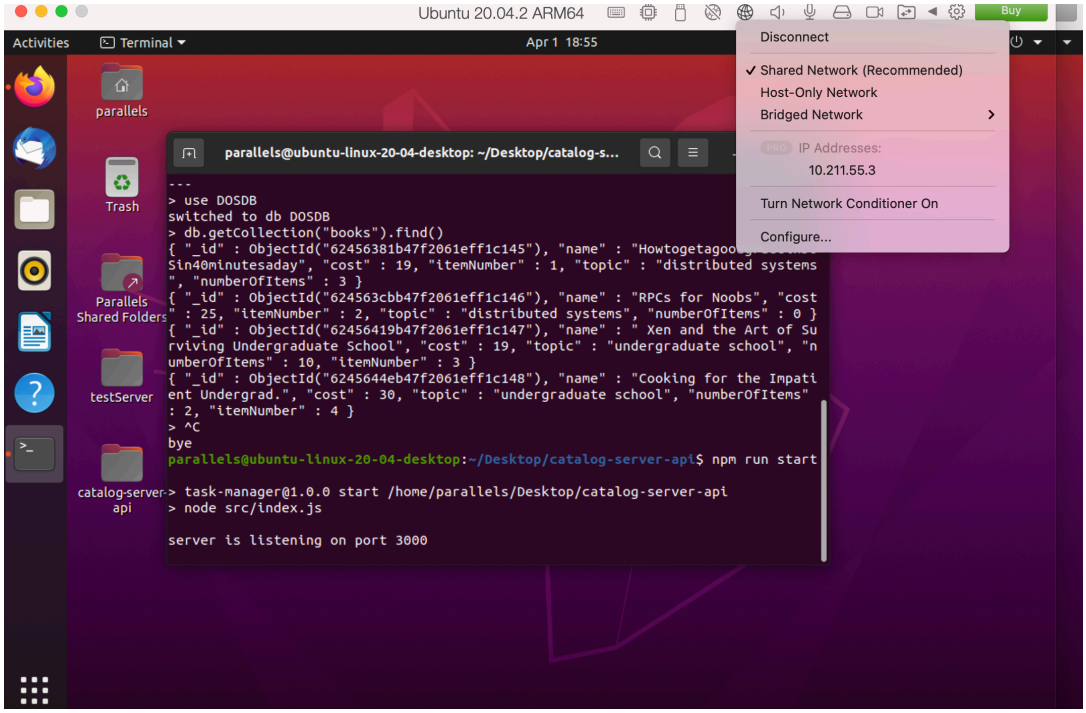
Figures from 1 to 14 describe the states of the servers and VMs:



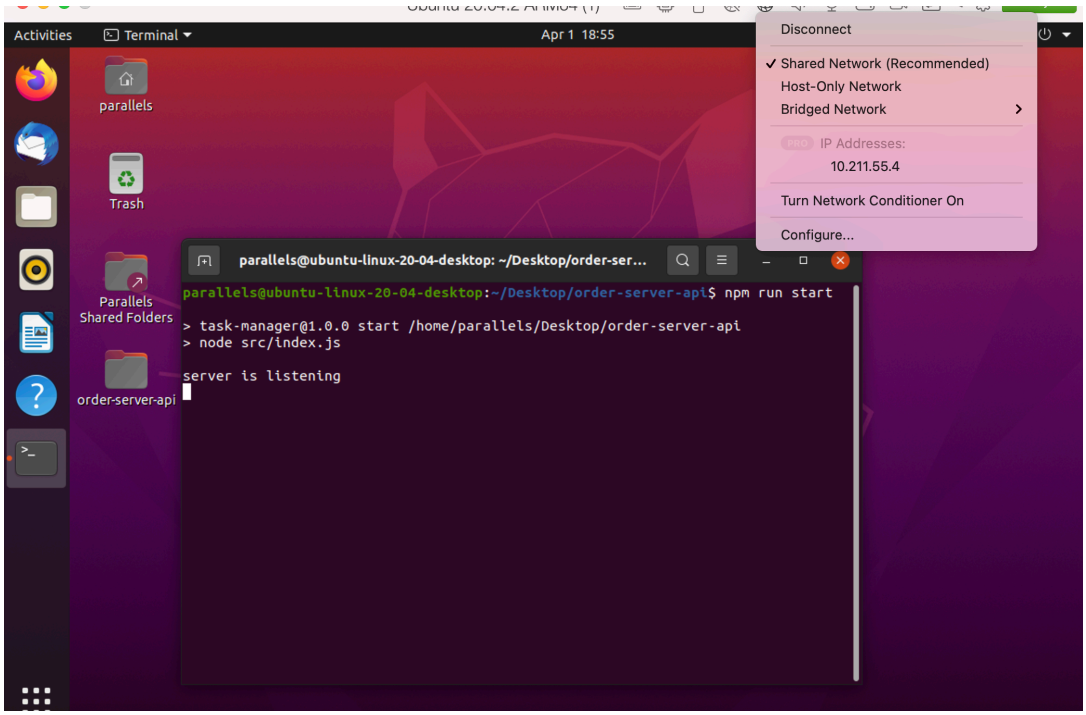
Figure(1): Postman sends request to the frontend server, express is listening on port 3002



Figure(2): on the first VM, I have mongoDB, with a collection filled with the information of the four books



Figure(3): On the same VM, I have the catalog server running on port 3000. VM's IP address: 10.211.55.3



Figure(4): On the second VM, I have the order server running on port 3000, VM's IP address: 10.211.55.4

Trade-Offs:

Like any microservices application:
The deployment part for each service independently was easy to do. Also, the client knows nothing about what happens behind the scene and only communicates with one server which is the frontend server.
However, there may be an overhead due to the communication between servers, moreover, the frontend server can become a bottleneck to the system.

Possible improvements and extensions:

All of the project's requirements were implemented and tested, but if I wanted to make the system larger, there could be a level of authentication on the user level to protect the data, also there could be an admin who can add new books to the database.

Any cases which don't work correctly:

I believe I implemented all the cases required correctly and tested them.

How you can run my program:

1- You need to have a tool like Postman to send requests.
2- On your machine you have to install Node.js and npm, then you can run the frontend server code after running (npm install) to install all of the packages needed.
3-On the first VM (catalog server): you need to install MongoDB and get it running on port 27017, and fill it with books with the attributes mentioned in the requirements of the project, you'll also have to install Node.js, npm and NginX, then you have to modify the configuration file of NginX to define it as

a proxy server that redirects requests from port 80 to port 3000. You also may need to give this virtual machine the same IP address as mine: 10.211.55.3, otherwise, you'll need to change the IP address used in code in both the frontend server and the order server. Then you can run my catalog server code (also after running npm install to install the packages needed).
4- On the second VM (order server): you need to install Node.js, npm and NginX and configure NginX to work as proxy server to redirect requests to post 3000, you may give this the machine the IP: 10.211.55.4, then you can run the order server code (also after running npm install to install the packages needed).
5- Now, you can start sending requests to localhost:3002

Please notice I have uploaded another document of the describes the exact endpoints and shows the responses, please refer to it

I pushed the code for the three services on GitHub:
Catalog server: https://github.com/hayamikkawi/catalog-server-api.git
Order server: https://github.com/hayamikkawi/order-server-api.git
Frontend server: https://github.com/hayamikkawi/frontend-server.git