بــسم هللا الـــرحـمـن الـــرحـيـم

Lab 2 Turning the Bazar into an Amazon: Replication, Caching and Consistency

الطالبان:

مصطفى كمال سكر
11819630
مصطفى عادل حب رمان
11820449

دكتور المساق: د.سامر العرندي

In this Lab we add replication and caching so we have five tires (servers): one for front-end and two replicas each for the order and catalog servers.

1. Front end server.
2. Order server1.
3. Order server2.
4. Catalog server1.
5. Catalog server2.

In addition to that, we use in-memory cache integrated into the front-end server.

The front end server checks the cache first before it forwards the request to one of the catalog servers.

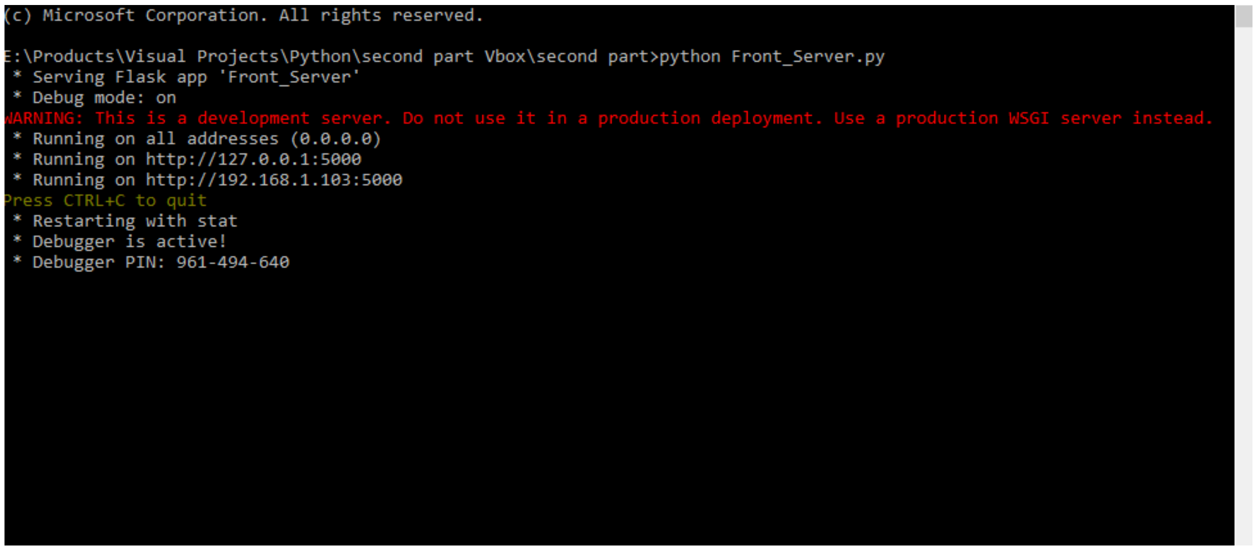
In the front end server we implement a round-robin as a load balancing algorithm that takes each incoming request and sends it to one of the replicas.

Below is a simplified explanation of each server:

1. Front end server: accept user requests and perform initial processing.
2. Order server: server that i can buy from the bookstore,server maintains a list of all orders received for the books.
3. Catalog server: small database of books.

Front end server: running on host OS

(windows)



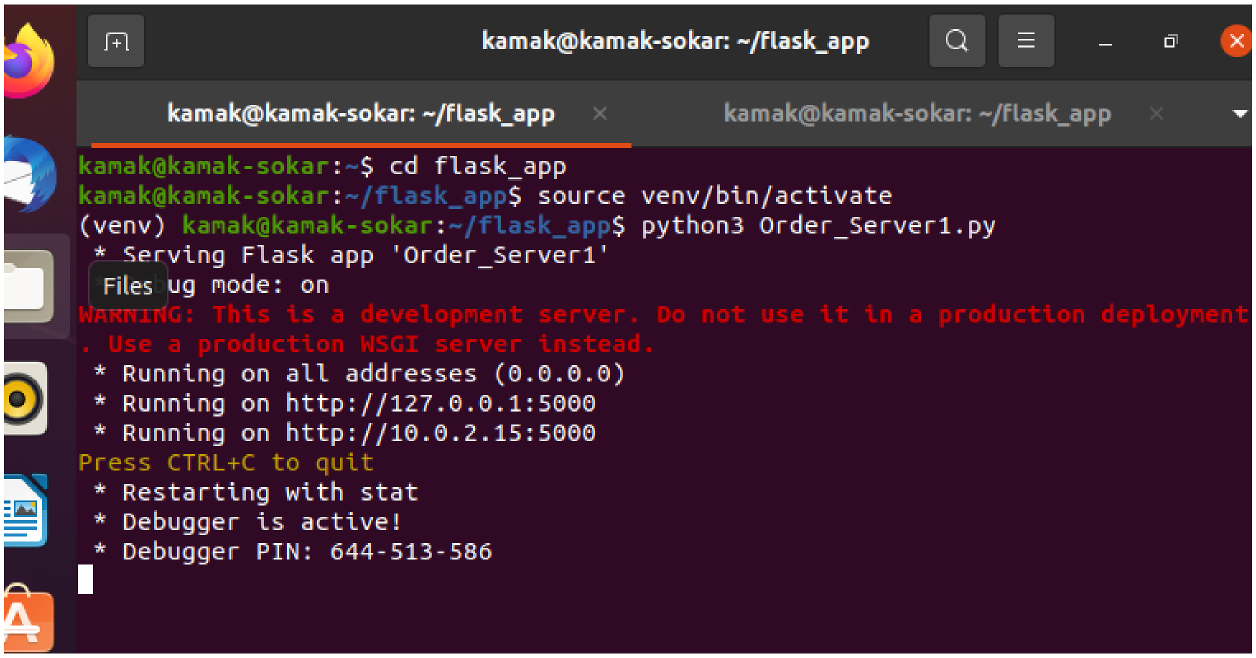
```
(c) Microsoft Corporation. All rights reserved.

E:\Products\Visual Projects\Python\second part Vbox\second part>python Front_Server.py
 * Serving Flask app 'Front_Server'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.103:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 961-494-640
```

Order servers: running on gest OS (Ubuntu)

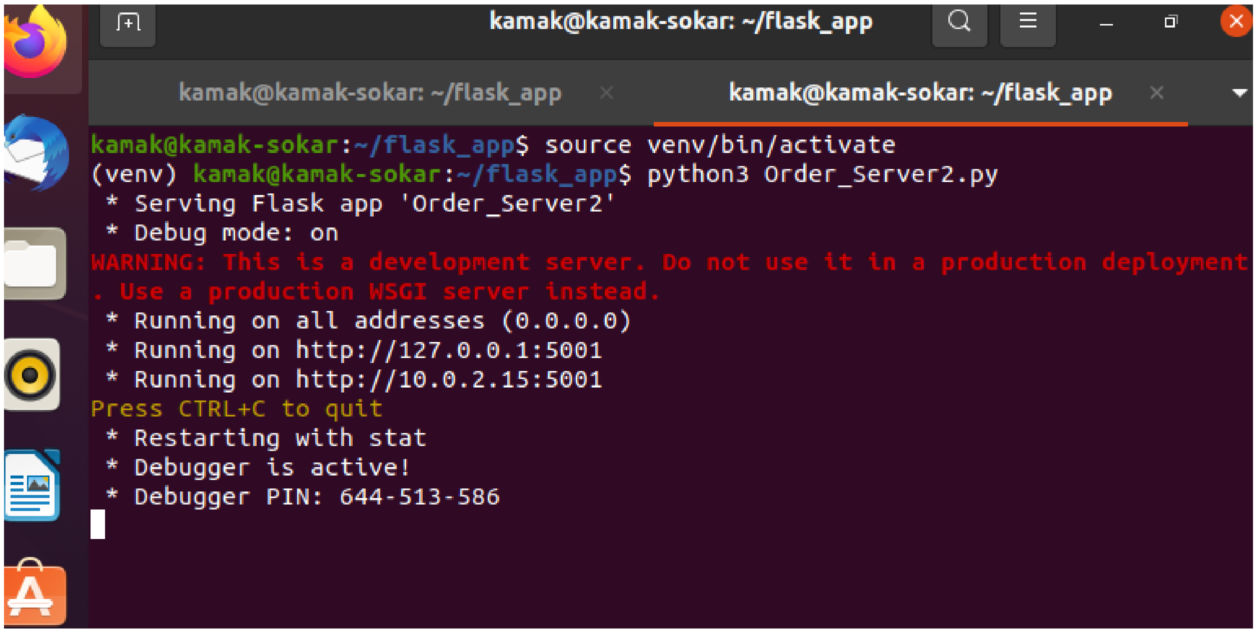
Two servers each one on different port number

Order server1:5000


kamak@kamak-sokar: ~/flask_app
kamak@kamak-sokar: ~/flask_app
kamak@kamak-sokar: ~/flask_app
Files

```
kamak@kamak-sokar:~$ cd flask_app
kamak@kamak-sokar:~/flask_app$ source venv/bin/activate
(venv) kamak@kamak-sokar:~/flask_app$ python3 Order_Server1.py
 * Serving Flask app 'Order_Server1'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment
. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://10.0.2.15:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 644-513-586
```

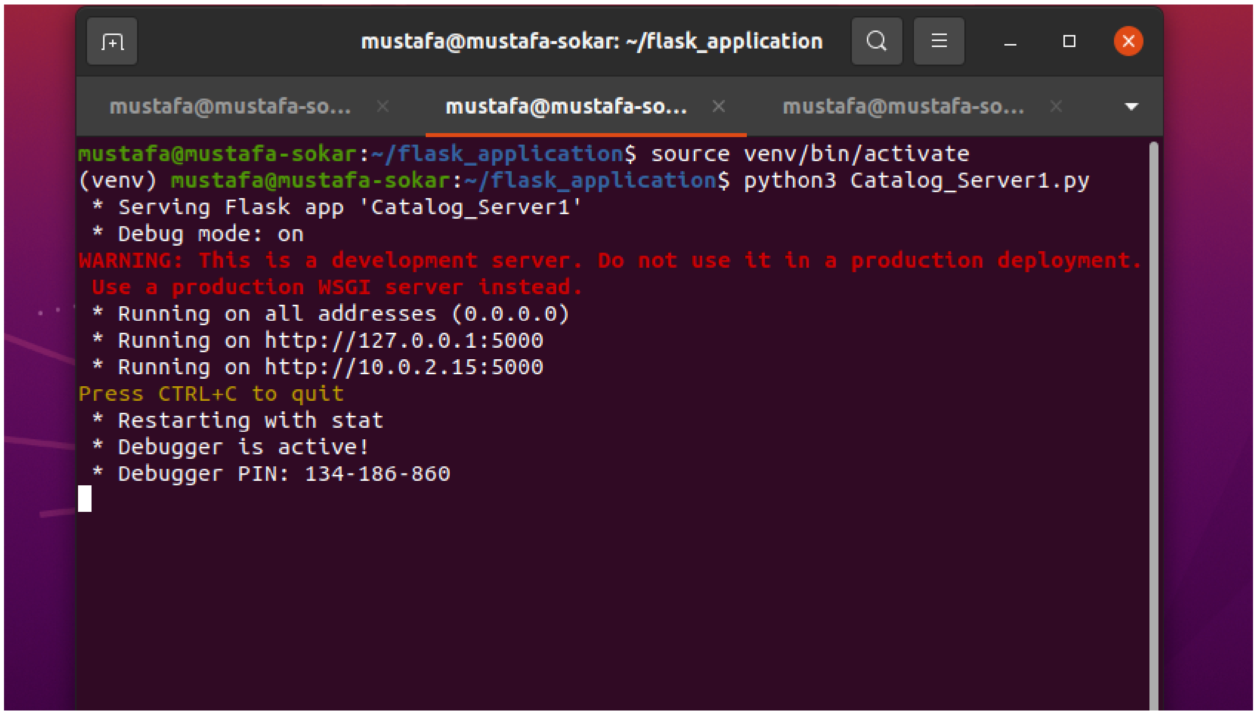
Order server2:5001


kamak@kamak-sokar: ~/flask_app
kamak@kamak-sokar: ~/flask_app
kamak@kamak-sokar: ~/flask_app

```
kamak@kamak-sokar:~/flask_app$ source venv/bin/activate
(venv) kamak@kamak-sokar:~/flask_app$ python3 Order_Server2.py
 * Serving Flask app 'Order_Server2'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment
. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5001
 * Running on http://10.0.2.15:5001
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 644-513-586
```

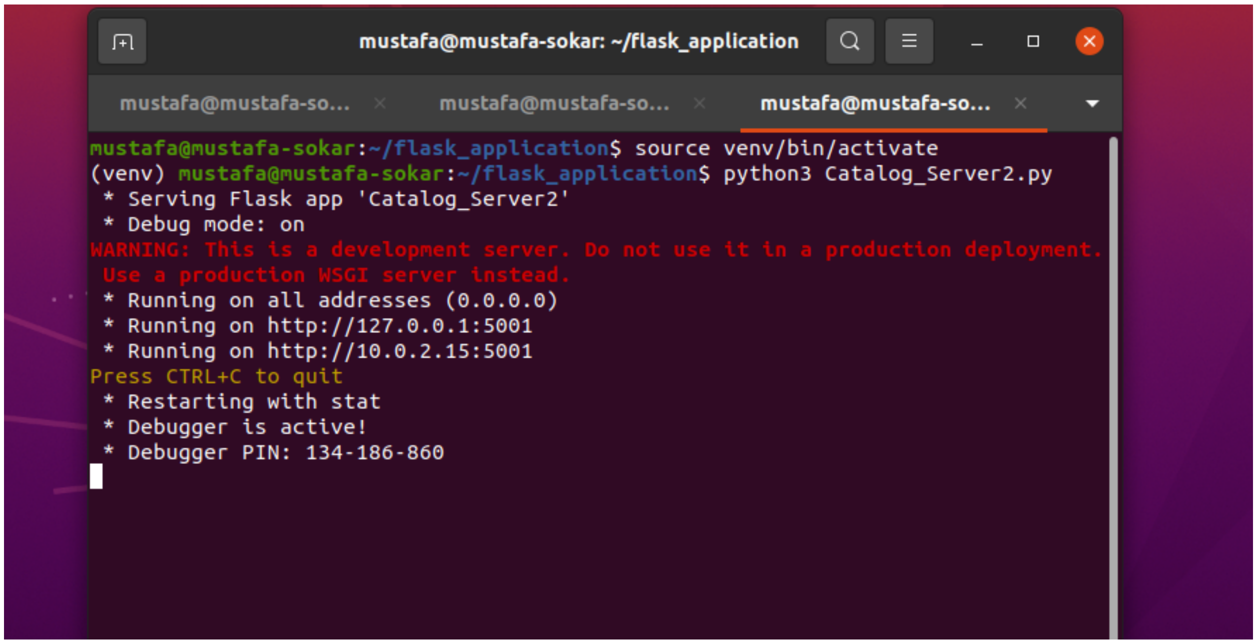
Catalog servers: running on another gest OS (Ubuntu)
Two servers each one on different port number

Catalog server1:5000



```
mustafa@mustafa-sokar:~/flask_application$ source venv/bin/activate
(venv) mustafa@mustafa-sokar:~/flask_application$ python3 Catalog_Server1.py
 * Serving Flask app 'Catalog_Server1'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
 Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://10.0.2.15:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 134-186-860
```

Catalog server2:5001

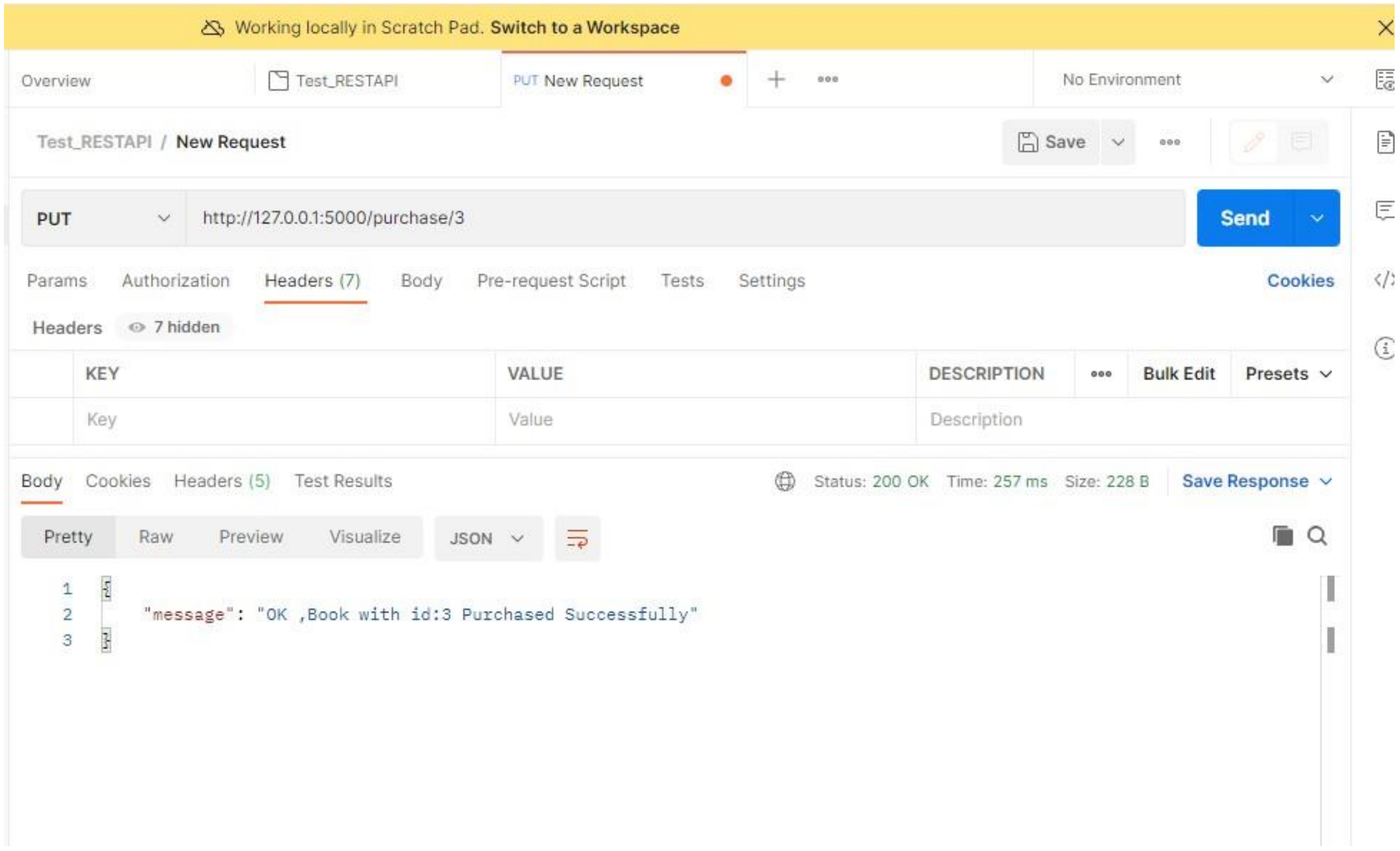


```
mustafa@mustafa-sokar:~/flask_application$ source venv/bin/activate
(venv) mustafa@mustafa-sokar:~/flask_application$ python3 Catalog_Server2.py
 * Serving Flask app 'Catalog_Server2'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
 Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5001
 * Running on http://10.0.2.15:5001
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 134-186-860
```

In this project we use flask framework (python language), when user need request from book store the front server take this request and if the request:

1. Search about book by topic: then the request go to the one of Catalog servers directly.
2. Information about specific book(id): then the request go to the one of Catalog servers directly.
3. Purchase: then the request go to the one of Order servers to verify that the item is in stock by querying the catalog server and then decrement the number of items in stock by one. The purchase request can fail if the item is out of stock.

When we send a purchase request to the front server by using postman:

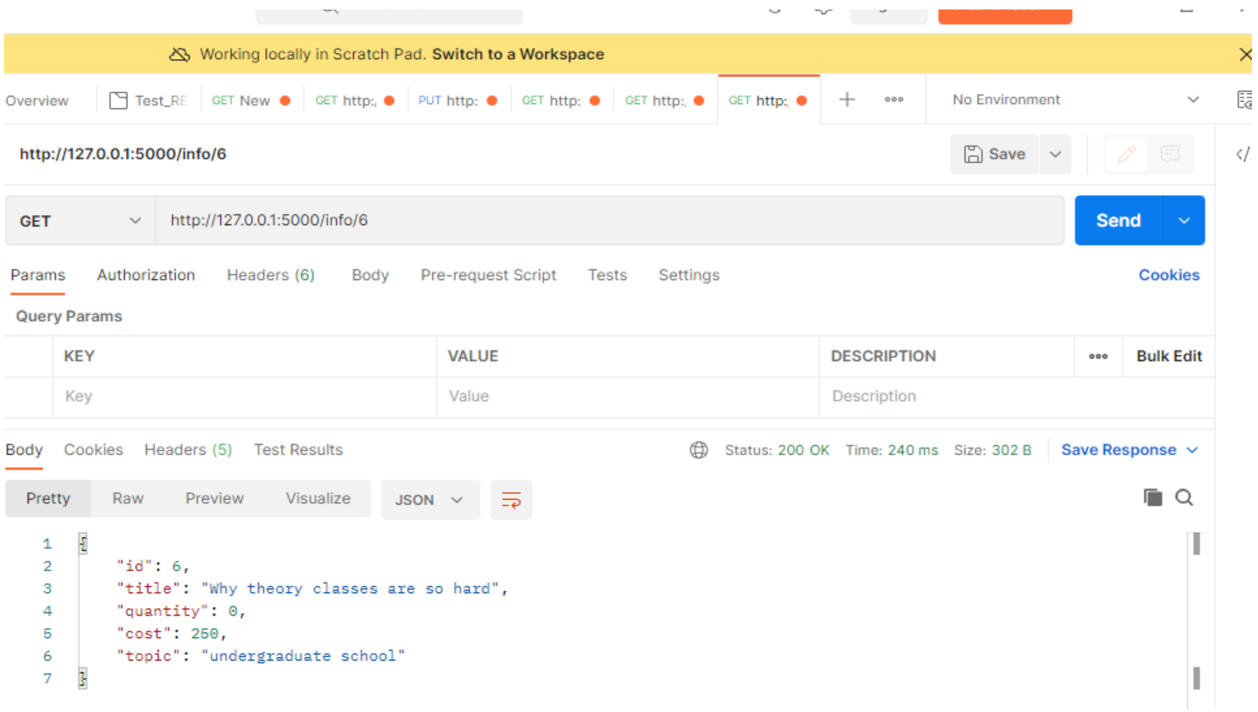


PUT http://127.0.0.1:5000/purchase/3

Status: 200 OK Time: 257 ms Size: 228 B

```
{
    "message": "OK ,Book with id:3 Purchased Successfully"
}
```

When we send a info request about book with id=6 , to the front server by using postman:



Working locally in Scratch Pad. **Switch to a Workspace**

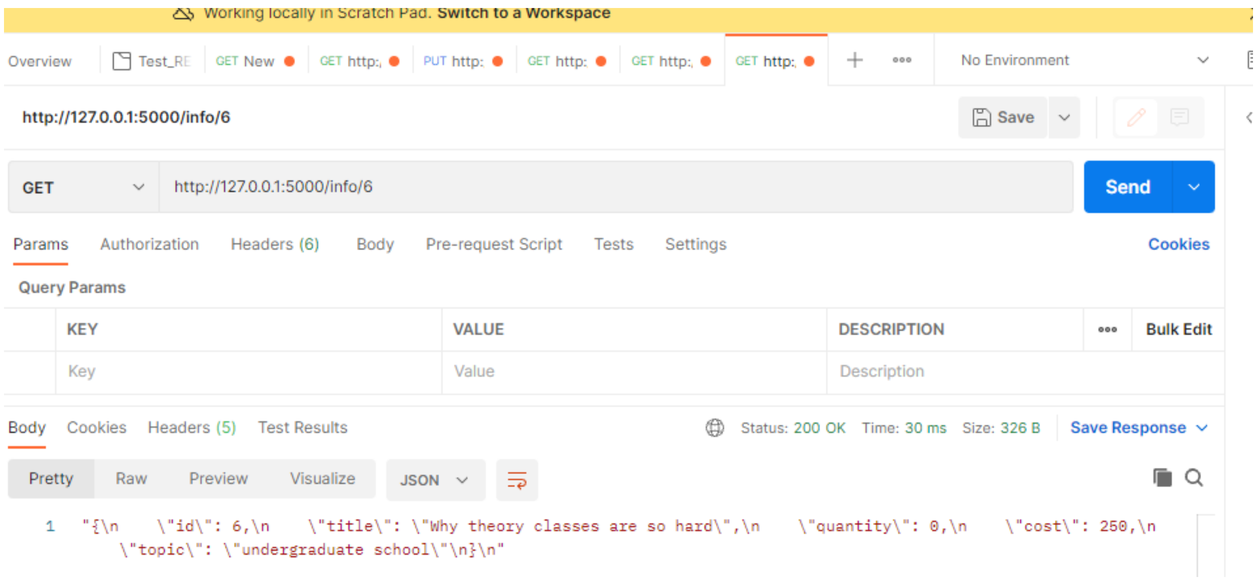
http://127.0.0.1:5000/info/6

GET http://127.0.0.1:5000/info/6

Status: 200 OK Time: 240 ms Size: 302 B

```
{
    "id": 6,
    "title": "Why theory classes are so hard",
    "quantity": 0,
    "cost": 250,
    "topic": "undergraduate school"
}
```

When we send the same info request about book with id=6 , to the front server by using postman we notice it find the query in the cache so it return the result from the cache:



Working locally in Scratch Pad. **Switch to a Workspace**

http://127.0.0.1:5000/info/6

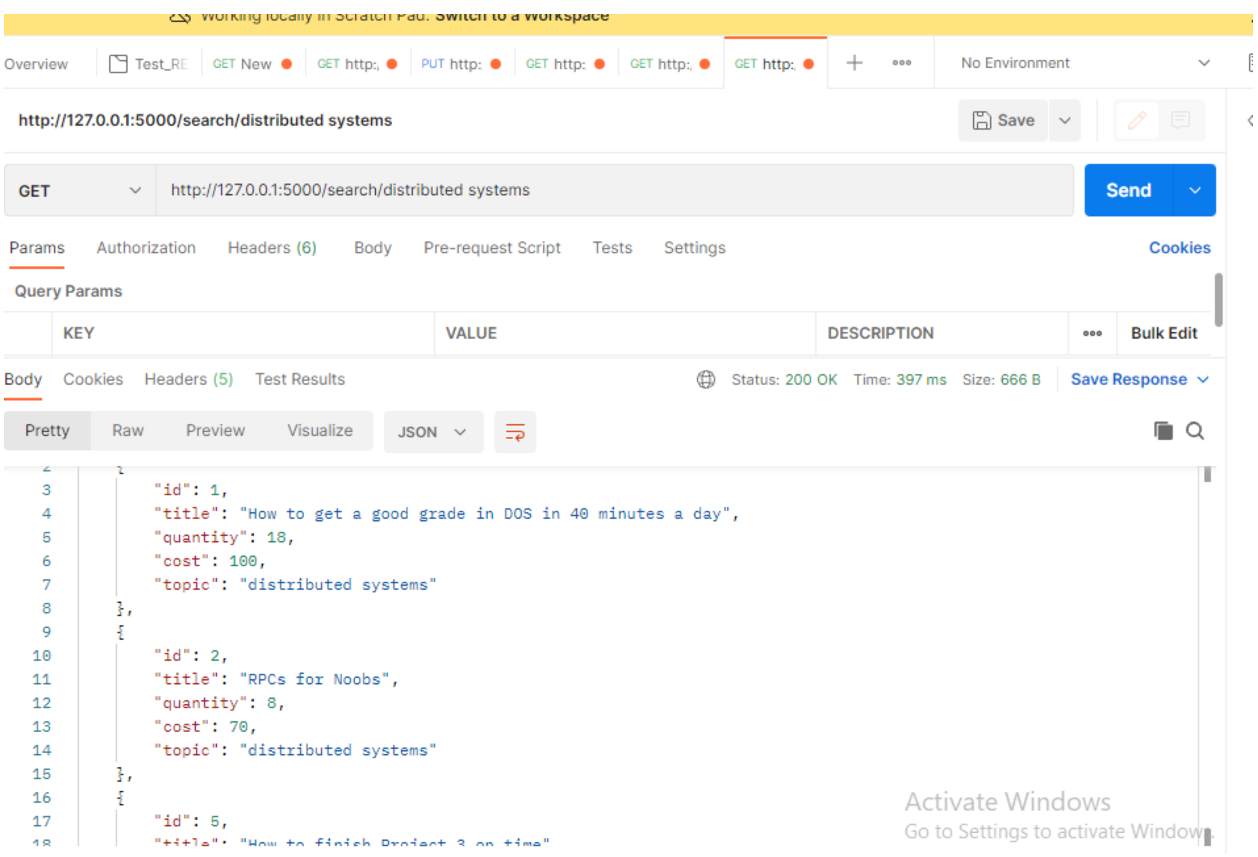
GET http://127.0.0.1:5000/info/6

Status: 200 OK Time: 30 ms Size: 326 B

```
"{\n    \"id\": 6,\n    \"title\": \"Why theory classes are so hard\",\n    \"quantity\": 0,\n    \"cost\": 250,\n
    \"topic\": \"undergraduate school\"\n}\n"
```

When we send a search request to the front server by using postman:

Front server checks the cache first not found so it forwards the request to the catalog server



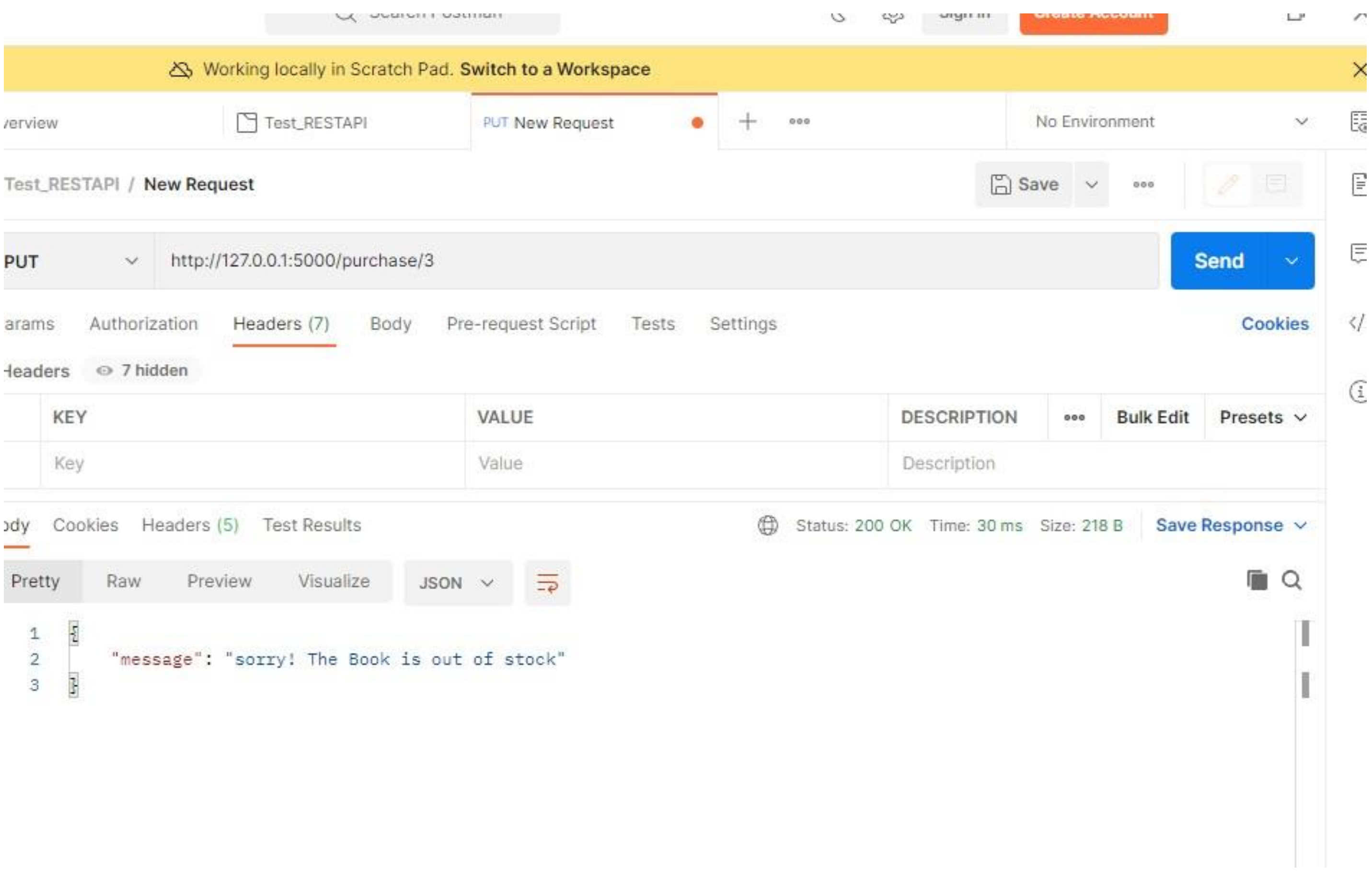
http://127.0.0.1:5000/search/distributed systems

GET http://127.0.0.1:5000/search/distributed systems

Status: 200 OK Time: 397 ms Size: 666 B

```
        "id": 1,
        "title": "How to get a good grade in DOS in 40 minutes a day",
        "quantity": 18,
        "cost": 100,
        "topic": "distributed systems"
    },
    {
        "id": 2,
        "title": "RPCs for Noobs",
        "quantity": 8,
        "cost": 70,
        "topic": "distributed systems"
    },
    {
        "id": 5,
        "title": "How to finish Project 3 on time"
```

1. When we send a purchase request to the front server by using postman , but the request fail because item is out of stock:



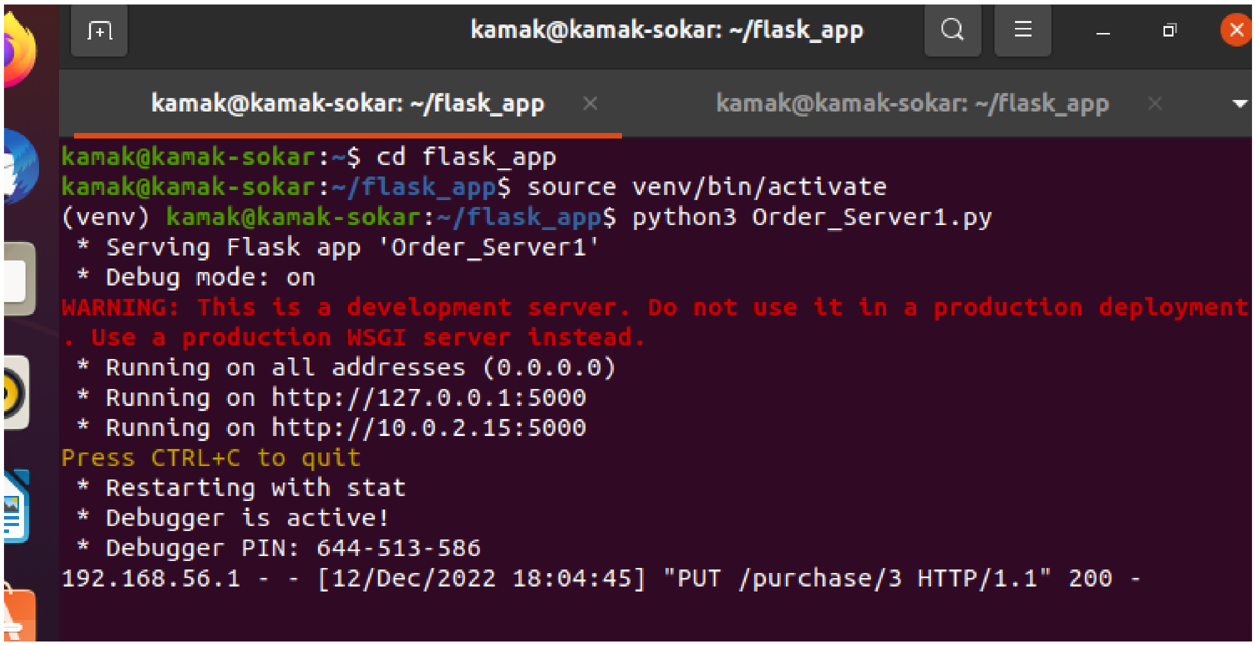
PUT http://127.0.0.1:5000/purchase/3

Status: 200 OK Time: 30 ms Size: 218 B

```
{
    "message": "sorry! The Book is out of stock"
}
```

And this receiving requests in servers when send purchase request:

Order server1:recive request from front server

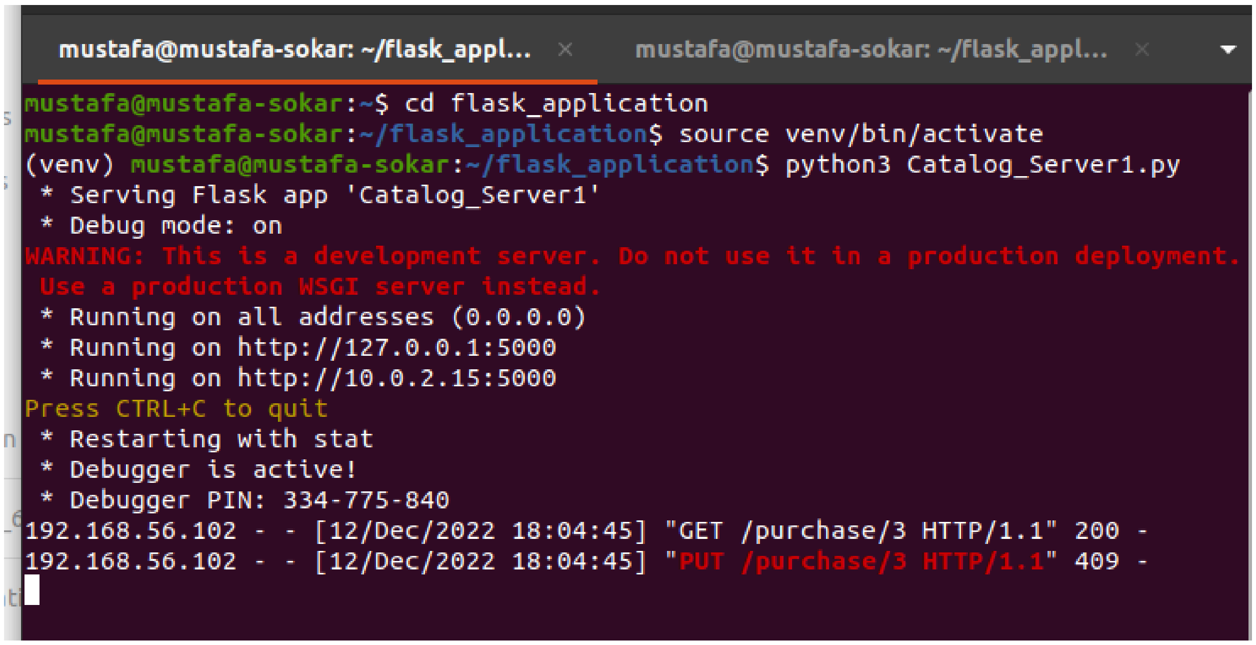


kamak@kamak-sokar: ~/flask_app
kamak@kamak-sokar: ~/flask_app
kamak@kamak-sokar: ~/flask_app

```
kamak@kamak-sokar:~$ cd flask_app
kamak@kamak-sokar:~/flask_app$ source venv/bin/activate
(venv) kamak@kamak-sokar:~/flask_app$ python3 Order_Server1.py
 * Serving Flask app 'Order_Server1'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment
. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://10.0.2.15:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 644-513-586
192.168.56.1 - - [12/Dec/2022 18:04:45] "PUT /purchase/3 HTTP/1.1" 200 -
```

Order server sends 2 requests to Catalog server1:

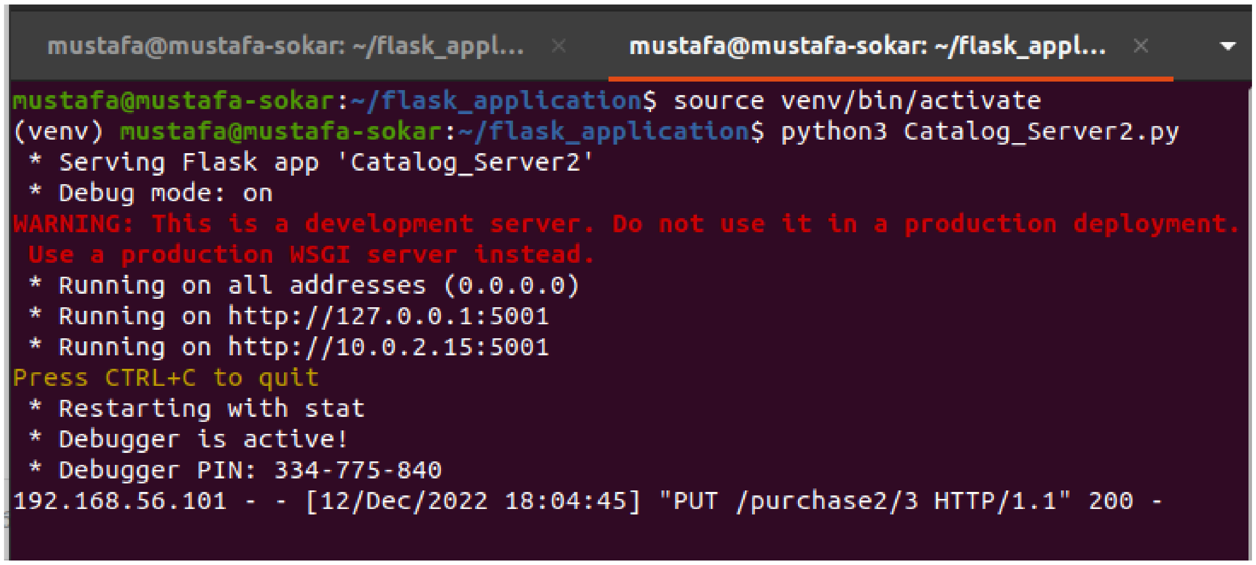
To verify that the item is in stock by querying the catalog server and then decrement the number of items in stock by one.



mustafa@mustafa-sokar: ~/flask_appl...
mustafa@mustafa-sokar: ~/flask_appl...

```
mustafa@mustafa-sokar:~$ cd flask_application
mustafa@mustafa-sokar:~/flask_application$ source venv/bin/activate
(venv) mustafa@mustafa-sokar:~/flask_application$ python3 Catalog_Server1.py
 * Serving Flask app 'Catalog_Server1'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
 Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://10.0.2.15:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 334-775-840
192.168.56.102 - - [12/Dec/2022 18:04:45] "GET /purchase/3 HTTP/1.1" 200 -
192.168.56.102 - - [12/Dec/2022 18:04:45] "PUT /purchase/3 HTTP/1.1" 409 -
```

For consistency catalog server1 send update request to catalog server 2

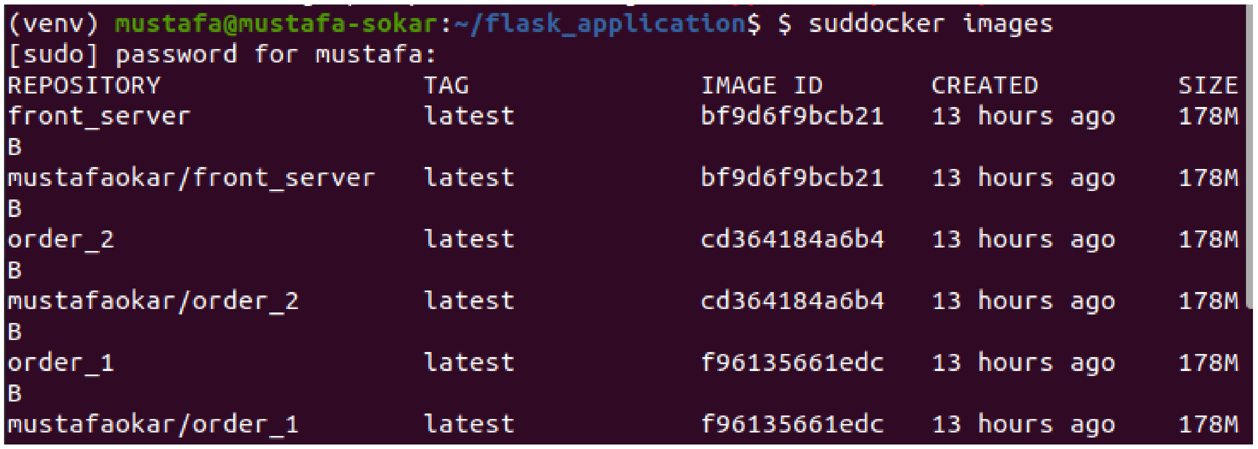

mustafa@mustafa-sokar: ~/flask_appl...
mustafa@mustafa-sokar: ~/flask_appl...

```
mustafa@mustafa-sokar:~/flask_application$ source venv/bin/activate
(venv) mustafa@mustafa-sokar:~/flask_application$ python3 Catalog_Server2.py
 * Serving Flask app 'Catalog_Server2'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
 Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5001
 * Running on http://10.0.2.15:5001
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 334-775-840
192.168.56.101 - - [12/Dec/2022 18:04:45] "PUT /purchase2/3 HTTP/1.1" 200 -
```

Part 2:Docker imge

We each component and package it as a container/ image

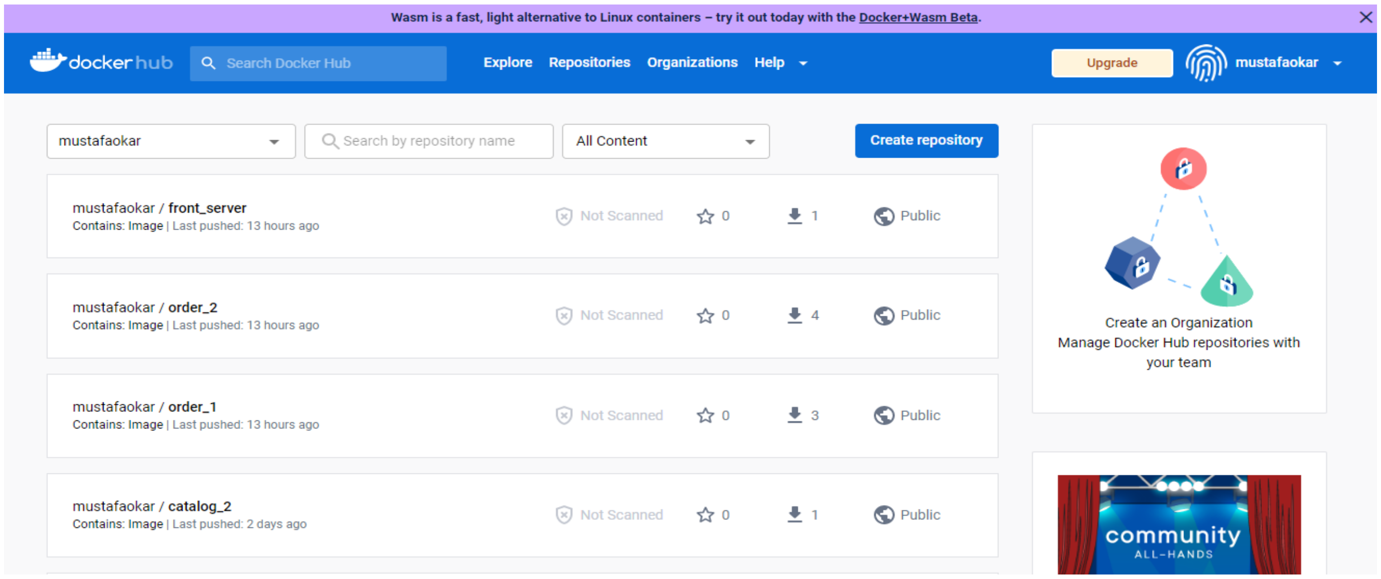
1-Write the code
2-Write the Docker file.
3-Build the Docker image. Using this command
Cmd:`sudo docker build --tag mustafaokar/front_server .`



```
(venv) mustafa@mustafa-sokar:~/flask_application$ $ suddocker images
[sudo] password for mustafa:
REPOSITORY                 TAG                 IMAGE ID            CREATED             SIZE
front_server               latest              bf9d6f9bcb21        13 hours ago        178M
B
mustafaokar/front_server   latest              bf9d6f9bcb21        13 hours ago        178M
B
order_2                    latest              cd364184a6b4        13 hours ago        178M
B
mustafaokar/order_2        latest              cd364184a6b4        13 hours ago        178M
B
order_1                    latest              f96135661edc        13 hours ago        178M
B
mustafaokar/order_1        latest              f96135661edc        13 hours ago        178M
```

4-Then we push the image to docker hub.
Cmd:`sudo docker push mustafaokar/front_server`

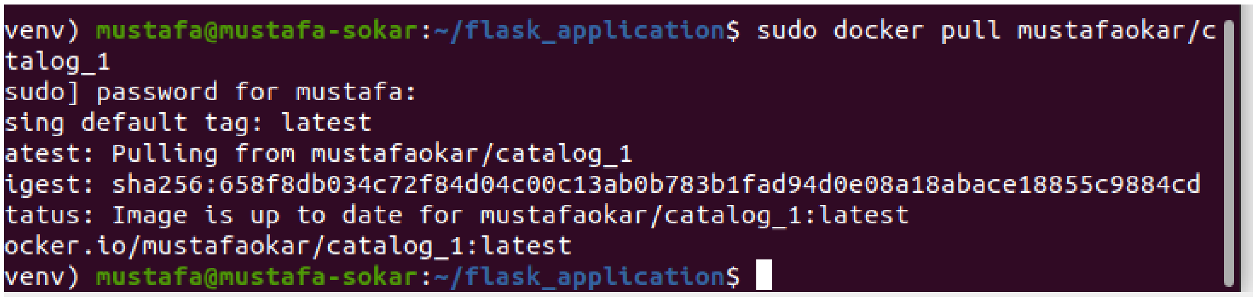
Docker images for all servers: you can pull it and run it:


Wasm is a fast, light alternative to Linux containers – try it out today with the Docker+Wasm Beta.
docker hub
Search Docker Hub
Explore
Repositories
Organizations
Help
Upgrade
mustafaokar
All Content
Search by repository name
Create repository
Create an Organization
Manage Docker Hub repositories with your team
community ALL-HANDS

mustafaokar / front_server Contains: Image | Last pushed: 13 hours ago	Not Scanned	0	1	Public
mustafaokar / order_2 Contains: Image | Last pushed: 13 hours ago	Not Scanned	0	4	Public
mustafaokar / order_1 Contains: Image | Last pushed: 13 hours ago	Not Scanned	0	3	Public
mustafaokar / catalog_2 Contains: Image | Last pushed: 2 days ago	Not Scanned	0	1	Public

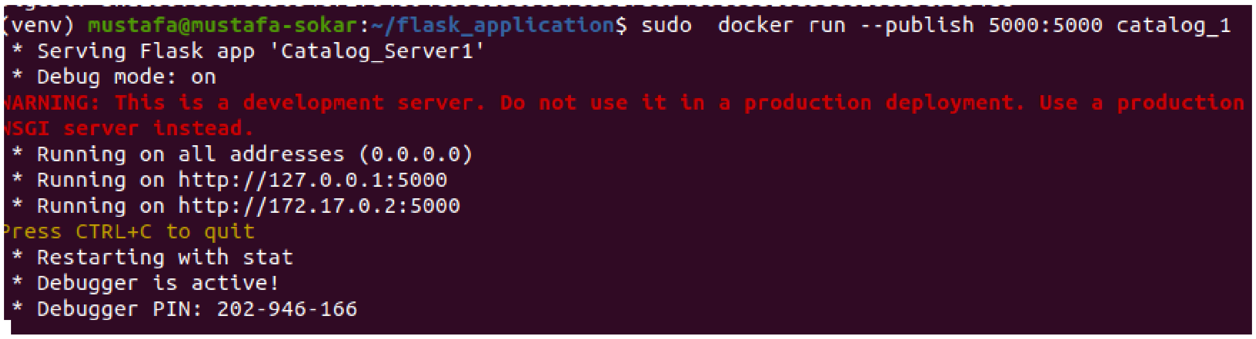
5- Pull the image you want on your machine from docker hub. For example:

Cmd:sudo docker pull mustafaokar/catalog_1



```
venv) mustafa@mustafa-sokar:~/flask_application$ sudo docker pull mustafaokar/c
talog_1
sudo] password for mustafa:
sing default tag: latest
atest: Pulling from mustafaokar/catalog_1
igest: sha256:658f8db034c72f84d04c00c13ab0b783b1fad94d0e08a18abace18855c9884cd
tatus: Image is up to date for mustafaokar/catalog_1:latest
ocker.io/mustafaokar/catalog_1:latest
venv) mustafa@mustafa-sokar:~/flask_application$
```

6-Then run the image:



```
(venv) mustafa@mustafa-sokar:~/flask_application$ sudo  docker run --publish 5000:5000 catalog_1
 * Serving Flask app 'Catalog_Server1'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production
WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 202-946-166
```