COMPSCI 677 Spring 2022

Lab 3: Caching, Replication and Fault Tolerance

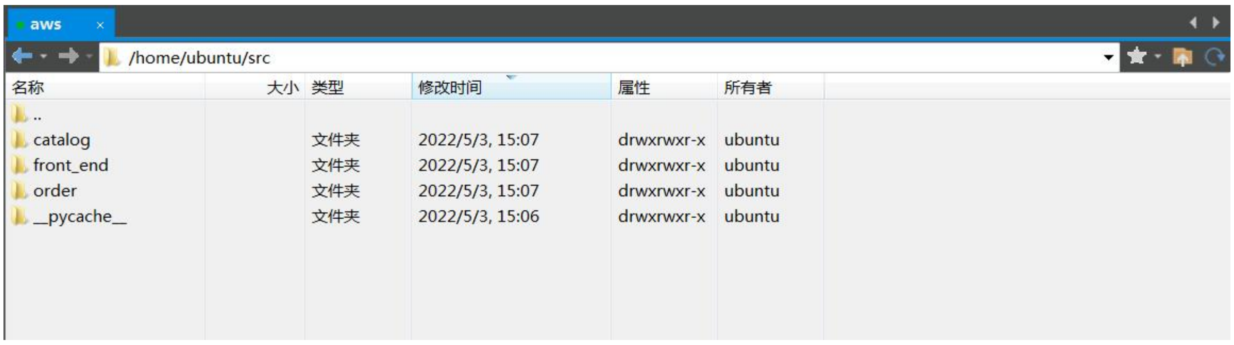
Team Members: Maoqin Zhu, Yixiang Zhang

Tutorial & Output Screenshots

1. Server Startup Screenshots

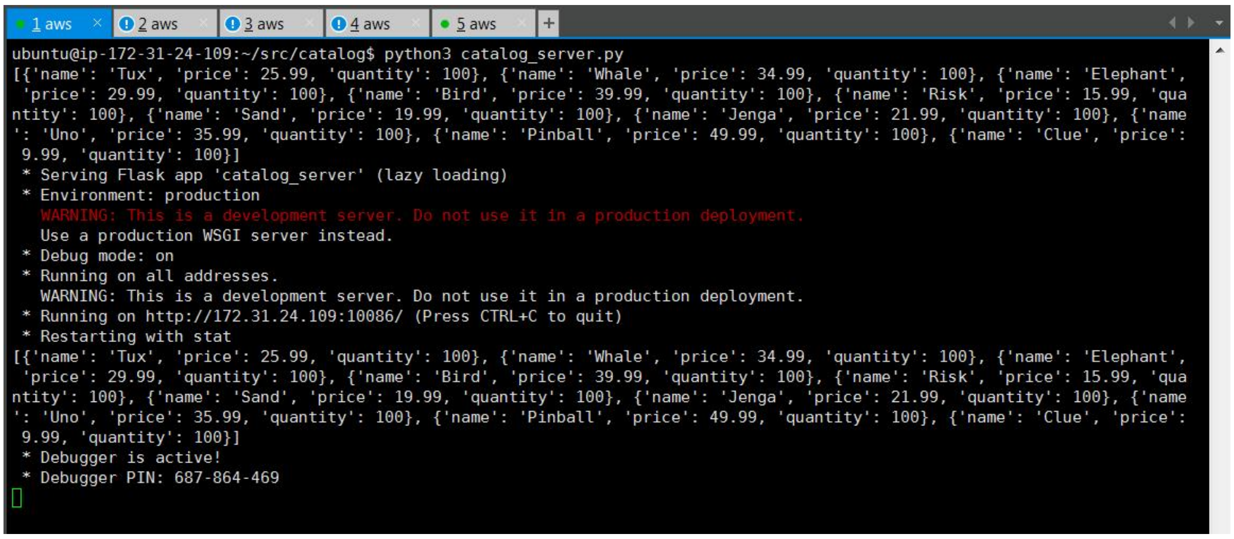
AWS EC2 Inscetance View- First of all, we should deploy our online application on AWS. Please checkout **the deployment tutorial** in “evaluation” file. After accessing our EC2 instance and upload source code via SSH, the online application are running as follows:

XShell View:



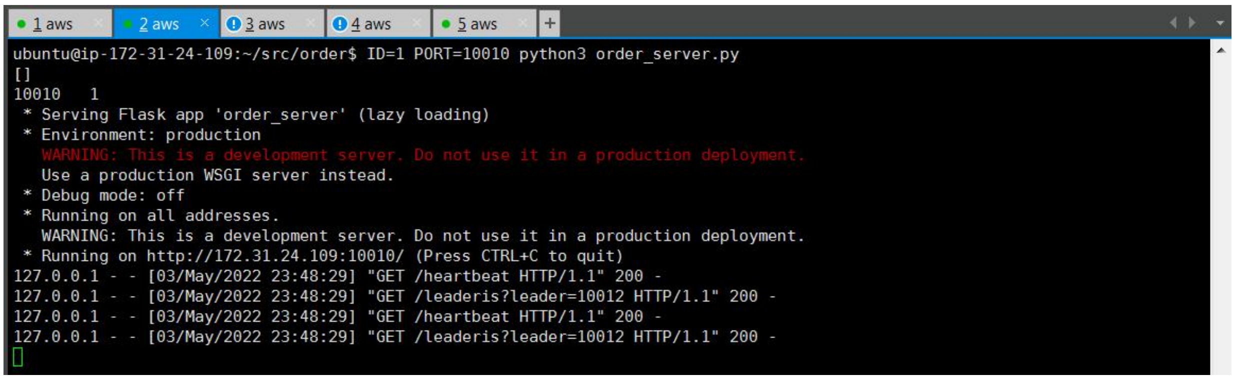
名称	大小	类型	修改时间	属性	所有者
..					
catalog		文件夹	2022/5/3, 15:07	drwxrwxr-x	ubuntu
front_end		文件夹	2022/5/3, 15:07	drwxrwxr-x	ubuntu
order		文件夹	2022/5/3, 15:07	drwxrwxr-x	ubuntu
__pycache__		文件夹	2022/5/3, 15:06	drwxrwxr-x	ubuntu

Terminal 1- Catalog Server:



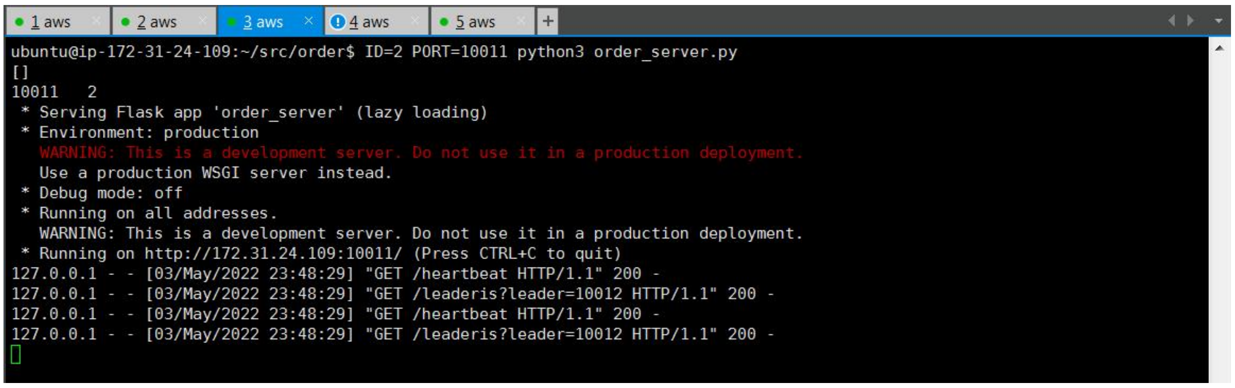
```
ubuntu@ip-172-31-24-109:~/src/catalog$ python3 catalog_server.py
[{'name': 'Tux', 'price': 25.99, 'quantity': 100}, {'name': 'Whale', 'price': 34.99, 'quantity': 100}, {'name': 'Elephant',
 'price': 29.99, 'quantity': 100}, {'name': 'Bird', 'price': 39.99, 'quantity': 100}, {'name': 'Risk', 'price': 15.99, 'qua
ntity': 100}, {'name': 'Sand', 'price': 19.99, 'quantity': 100}, {'name': 'Jenga', 'price': 21.99, 'quantity': 100}, {'name
': 'Uno', 'price': 35.99, 'quantity': 100}, {'name': 'Pinball', 'price': 49.99, 'quantity': 100}, {'name': 'Clue', 'price':
 9.99, 'quantity': 100}]
 * Serving Flask app 'catalog_server' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.31.24.109:10086/ (Press CTRL+C to quit)
 * Restarting with stat
[{'name': 'Tux', 'price': 25.99, 'quantity': 100}, {'name': 'Whale', 'price': 34.99, 'quantity': 100}, {'name': 'Elephant',
 'price': 29.99, 'quantity': 100}, {'name': 'Bird', 'price': 39.99, 'quantity': 100}, {'name': 'Risk', 'price': 15.99, 'qua
ntity': 100}, {'name': 'Sand', 'price': 19.99, 'quantity': 100}, {'name': 'Jenga', 'price': 21.99, 'quantity': 100}, {'name
': 'Uno', 'price': 35.99, 'quantity': 100}, {'name': 'Pinball', 'price': 49.99, 'quantity': 100}, {'name': 'Clue', 'price':
 9.99, 'quantity': 100}]
 * Debugger is active!
 * Debugger PIN: 687-864-469
```

Terminal 2- Order Server1:



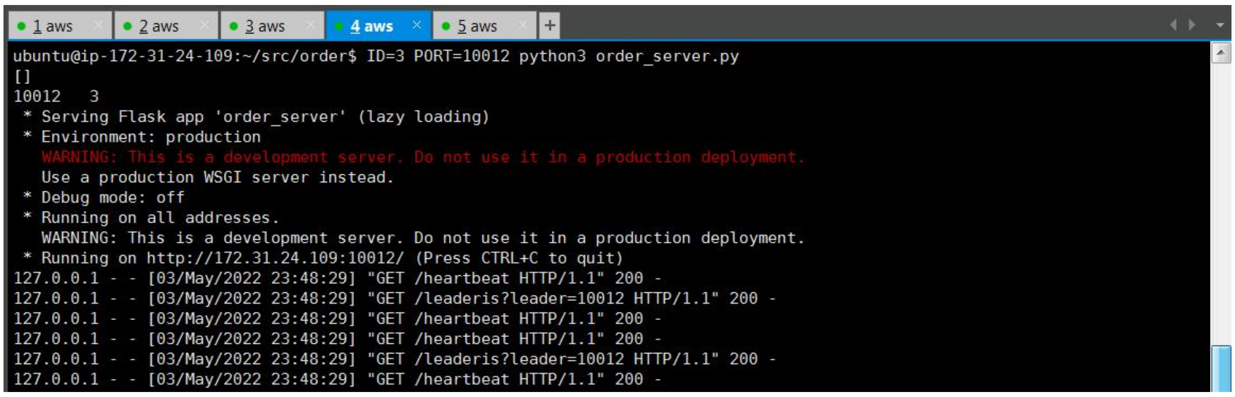
```
ubuntu@ip-172-31-24-109:~/src/order$ ID=1 PORT=10010 python3 order_server.py
[]
10010   1
 * Serving Flask app 'order_server' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.31.24.109:10010/ (Press CTRL+C to quit)
127.0.0.1 - - [03/May/2022 23:48:29] "GET /heartbeat HTTP/1.1" 200 -
127.0.0.1 - - [03/May/2022 23:48:29] "GET /leaderis?leader=10012 HTTP/1.1" 200 -
127.0.0.1 - - [03/May/2022 23:48:29] "GET /heartbeat HTTP/1.1" 200 -
127.0.0.1 - - [03/May/2022 23:48:29] "GET /leaderis?leader=10012 HTTP/1.1" 200 -
```

Terminal 3- Order Server2:



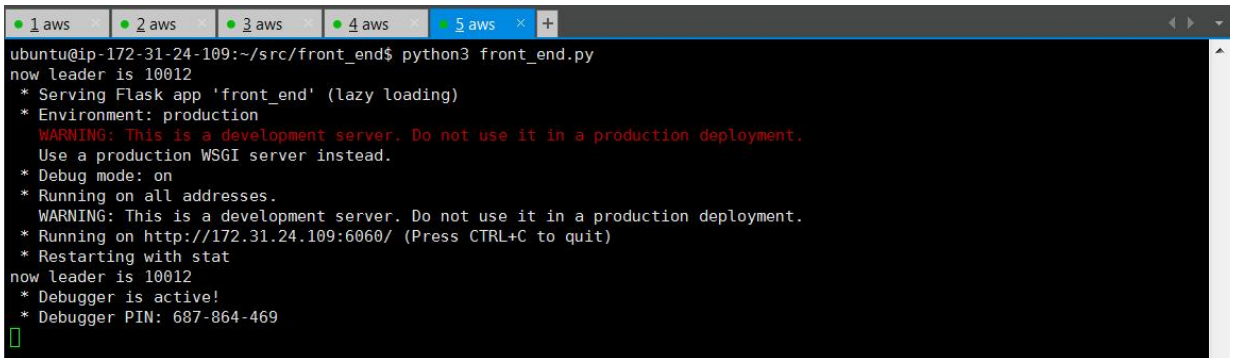
```
ubuntu@ip-172-31-24-109:~/src/order$ ID=2 PORT=10011 python3 order_server.py
[]
10011   2
 * Serving Flask app 'order_server' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.31.24.109:10011/ (Press CTRL+C to quit)
127.0.0.1 - - [03/May/2022 23:48:29] "GET /heartbeat HTTP/1.1" 200 -
127.0.0.1 - - [03/May/2022 23:48:29] "GET /leaderis?leader=10012 HTTP/1.1" 200 -
127.0.0.1 - - [03/May/2022 23:48:29] "GET /heartbeat HTTP/1.1" 200 -
127.0.0.1 - - [03/May/2022 23:48:29] "GET /leaderis?leader=10012 HTTP/1.1" 200 -
```

Terminal 4- Order Server3:



```
ubuntu@ip-172-31-24-109:~/src/order$ ID=3 PORT=10012 python3 order_server.py
[]
10012   3
 * Serving Flask app 'order_server' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.31.24.109:10012/ (Press CTRL+C to quit)
127.0.0.1 - - [03/May/2022 23:48:29] "GET /heartbeat HTTP/1.1" 200 -
127.0.0.1 - - [03/May/2022 23:48:29] "GET /leaderis?leader=10012 HTTP/1.1" 200 -
127.0.0.1 - - [03/May/2022 23:48:29] "GET /heartbeat HTTP/1.1" 200 -
127.0.0.1 - - [03/May/2022 23:48:29] "GET /heartbeat HTTP/1.1" 200 -
127.0.0.1 - - [03/May/2022 23:48:29] "GET /leaderis?leader=10012 HTTP/1.1" 200 -
127.0.0.1 - - [03/May/2022 23:48:29] "GET /heartbeat HTTP/1.1" 200 -
```

Terminal 5- Front End:

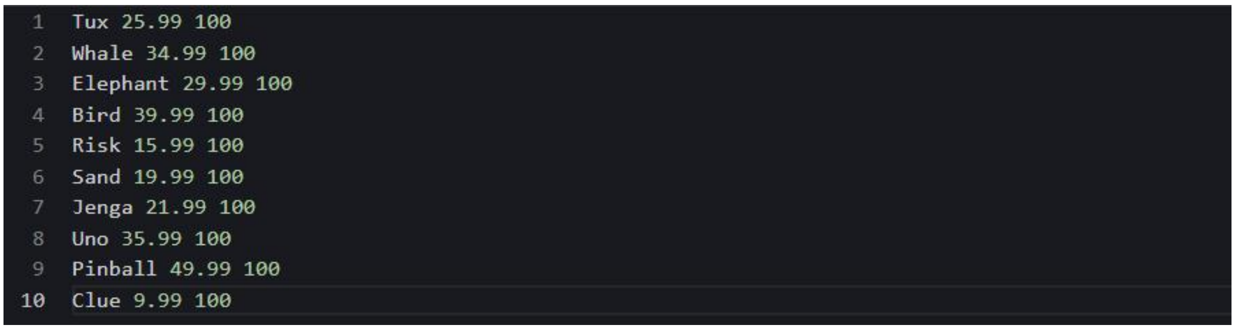


```
ubuntu@ip-172-31-24-109:~/src/front_end$ python3 front_end.py
now leader is 10012
 * Serving Flask app 'front_end' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.31.24.109:6060/ (Press CTRL+C to quit)
 * Restarting with stat
now leader is 10012
 * Debugger is active!
 * Debugger PIN: 687-864-469
```

2. Functional Test Output

Automated Testing- Looking at **"test_func.py"**, for different HTTP GET / HTTP POST, we created 19 test cases which correspond to 19 possible HTTP responses.

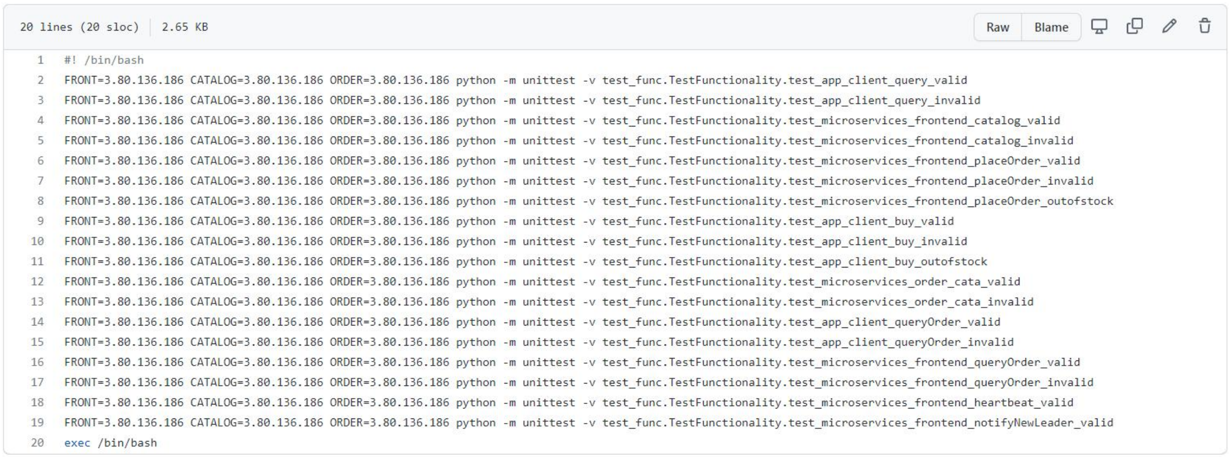
Notice that our test cases are effective only when database is in initial state, because expected response is configured statically in testing codes. Of course, you can also run your own test case simply by configuring request parameters and expected responses in the method. The initial state of database should be:



```
Tux 25.99 100
Whale 34.99 100
Elephant 29.99 100
Bird 39.99 100
Risk 15.99 100
Sand 19.99 100
Jenga 21.99 100
Uno 35.99 100
Pinball 49.99 100
Clue 9.99 100
```

Looking at **"test_func.sh"**, this shell file will help us run all the 19 test cases.

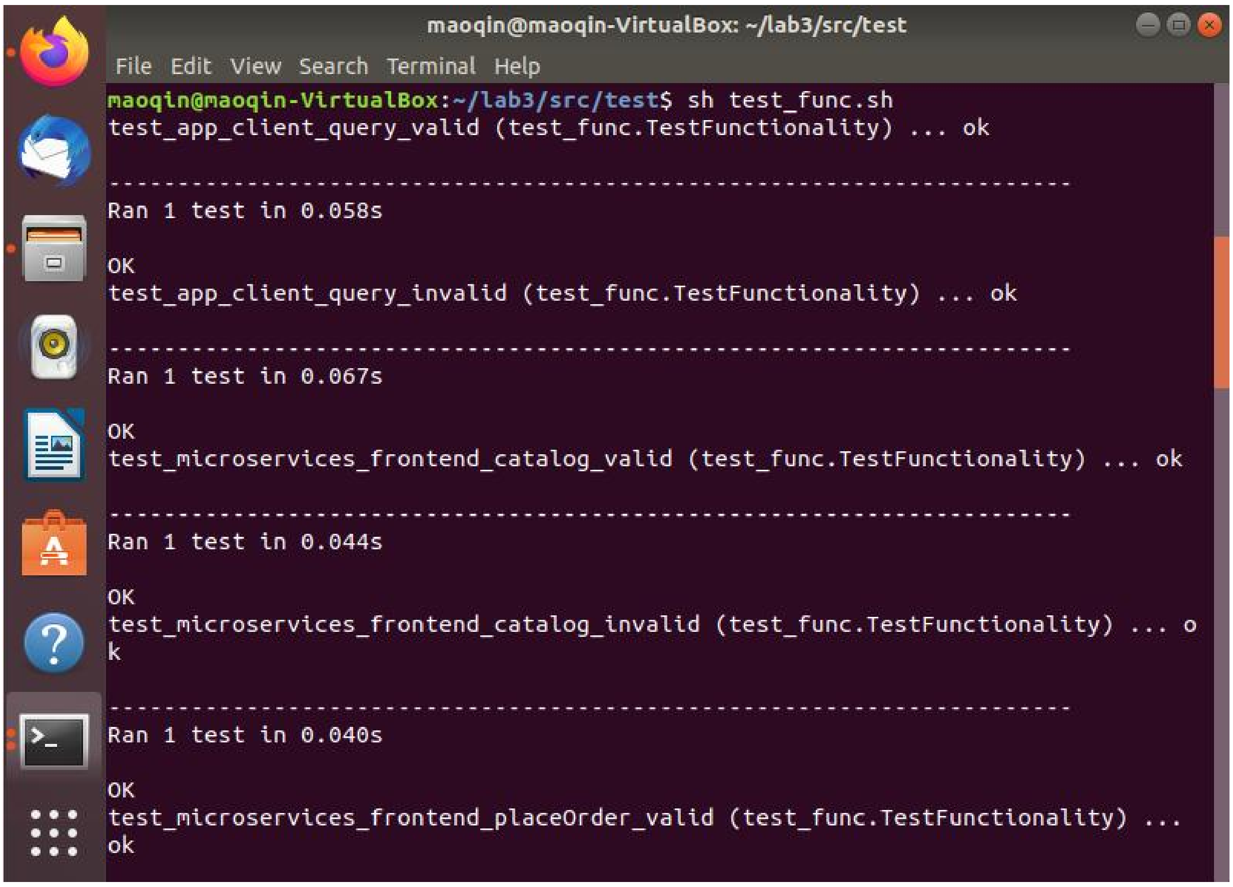
Notice that each time if you are running this shell, please configure those IP addresses(environment variables) manually. Thank you!!!



```
#! /bin/bash
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_app_client_query_valid
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_app_client_query_invalid
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_microservices_frontend_catalog_valid
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_microservices_frontend_catalog_invalid
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_microservices_frontend_placeOrder_valid
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_microservices_frontend_placeOrder_invalid
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_microservices_frontend_placeOrder_outofstock
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_app_client_buy_valid
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_app_client_buy_invalid
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_app_client_buy_outofstock
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_microservices_order_cata_valid
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_microservices_order_cata_invalid
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_app_client_queryOrder_valid
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_app_client_queryOrder_invalid
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_microservices_frontend_queryOrder_valid
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_microservices_frontend_queryOrder_invalid
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_microservices_frontend_heartbeat_valid
FRONT=3.80.136.186 CATALOG=3.80.136.186 ORDER=3.80.136.186 python -m unittest -v test_func.TestFunctionality.test_microservices_frontend_notifyNewLeader_valid
exec /bin/bash
```

Now, simply type the command: **$ sh test_func.sh**

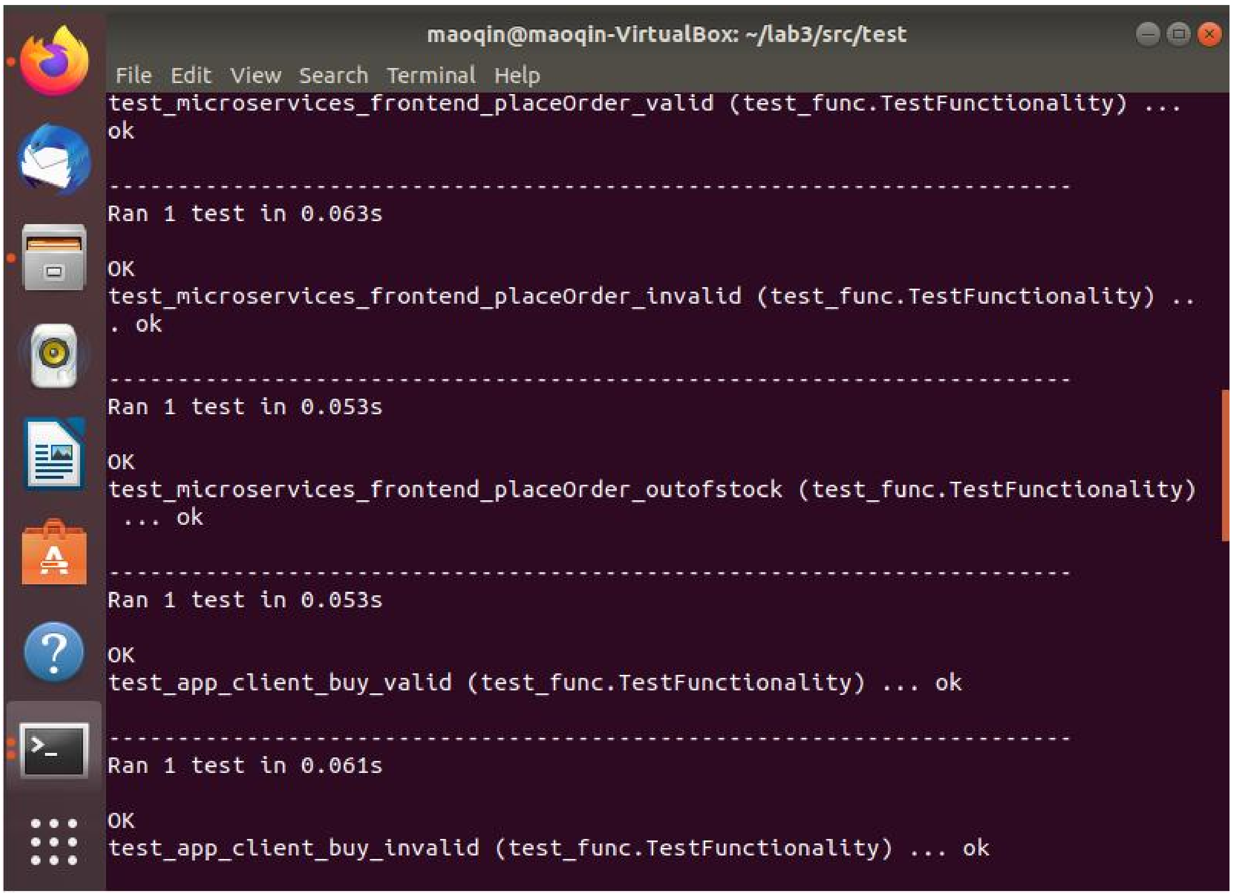
For each test case(valid/invalid requests), if our application or micro-services work correctly, Python unittest will tell **"ok"** on your terminal. As you can see, all the functionalities is working correctly as follows.



maoqin@maoqin-VirtualBox: ~/lab3/src/test

File Edit View Search Terminal Help

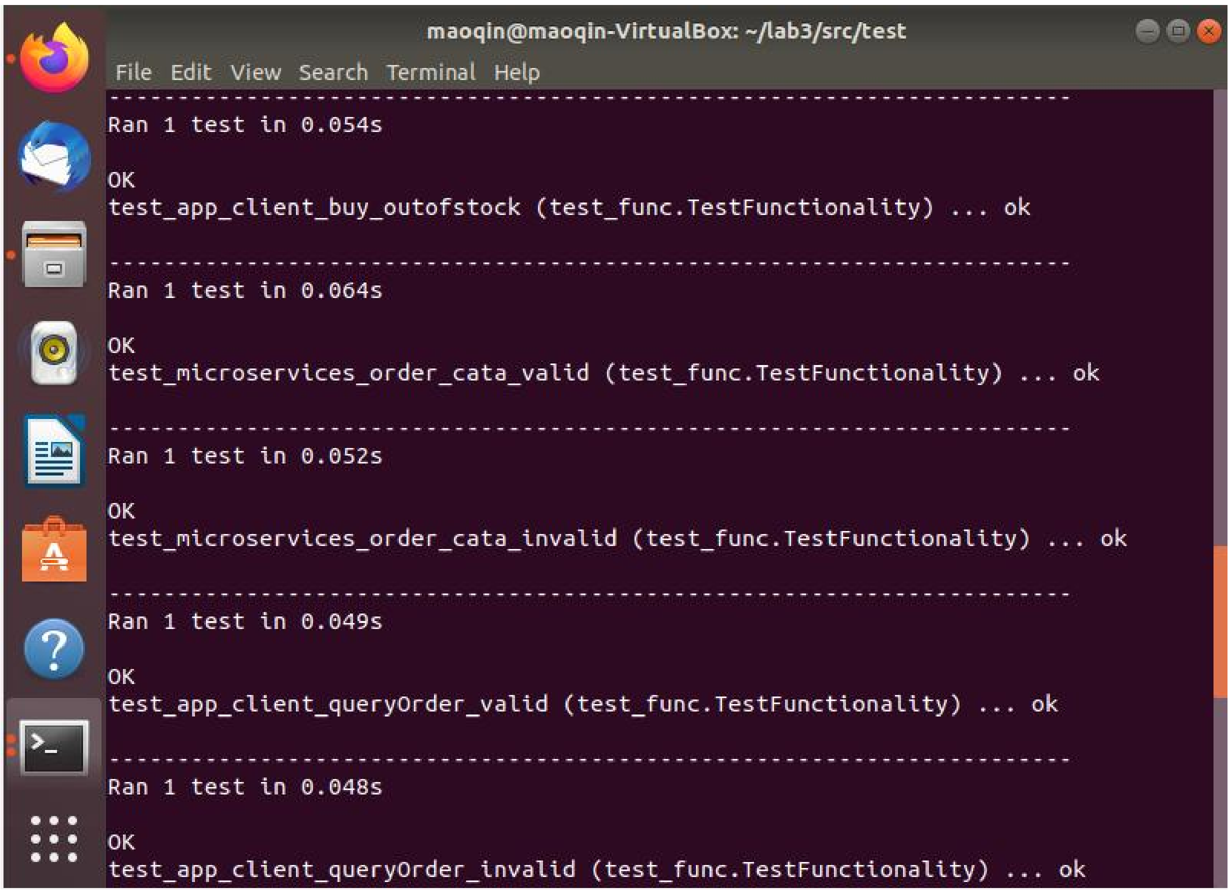
```
maoqin@maoqin-VirtualBox:~/lab3/src/test$ sh test_func.sh
test_app_client_query_valid (test_func.TestFunctionality) ... ok

----------------------------------------------------------------------
Ran 1 test in 0.058s

OK
test_app_client_query_invalid (test_func.TestFunctionality) ... ok

----------------------------------------------------------------------
Ran 1 test in 0.067s

OK
test_microservices_frontend_catalog_valid (test_func.TestFunctionality) ... ok

----------------------------------------------------------------------
Ran 1 test in 0.044s

OK
test_microservices_frontend_catalog_invalid (test_func.TestFunctionality) ... o
k

----------------------------------------------------------------------
Ran 1 test in 0.040s

OK
test_microservices_frontend_placeOrder_valid (test_func.TestFunctionality) ...
ok
```



maoqin@maoqin-VirtualBox: ~/lab3/src/test

File Edit View Search Terminal Help

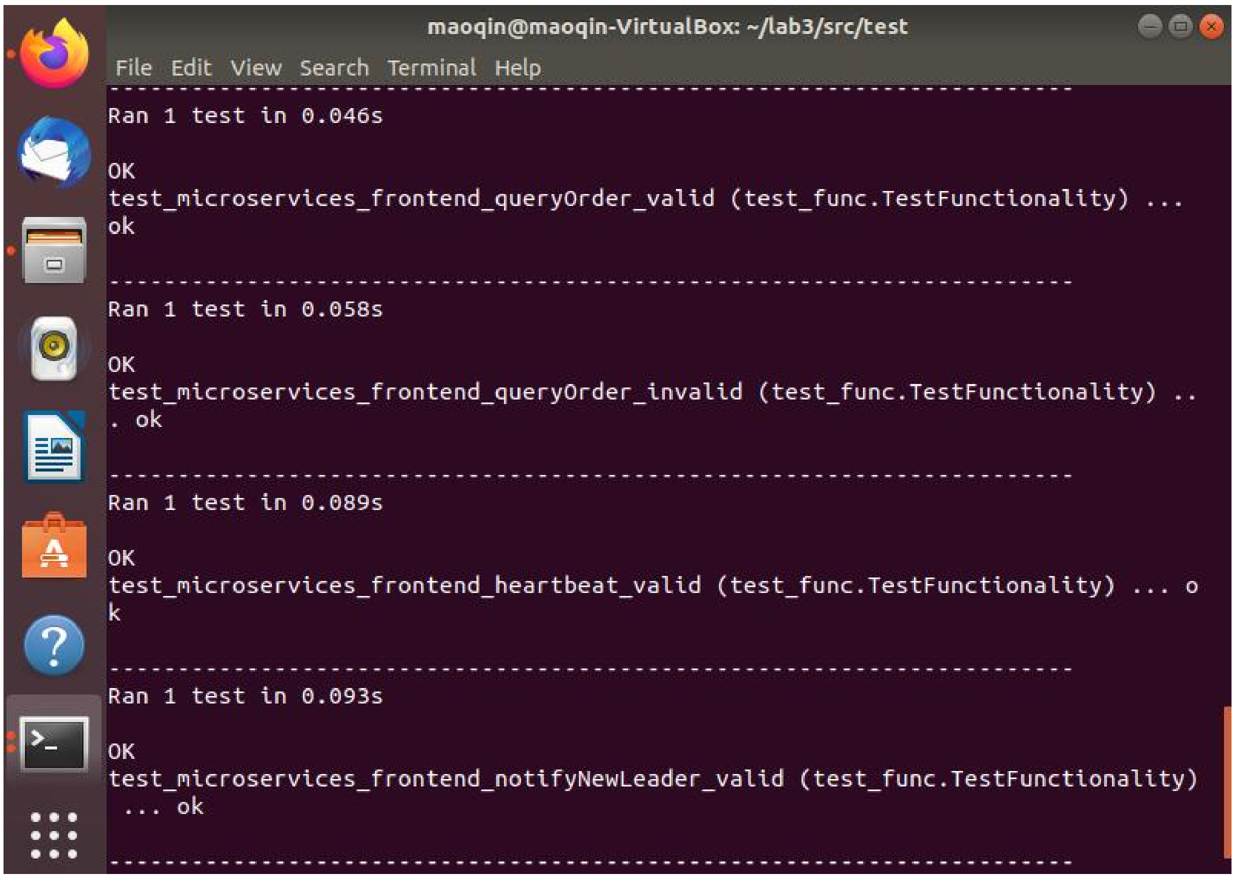
```
test_microservices_frontend_placeOrder_valid (test_func.TestFunctionality) ...
ok

----------------------------------------------------------------------
Ran 1 test in 0.063s

OK
test_microservices_frontend_placeOrder_invalid (test_func.TestFunctionality) ..
. ok

----------------------------------------------------------------------
Ran 1 test in 0.053s

OK
test_microservices_frontend_placeOrder_outofstock (test_func.TestFunctionality)
 ... ok

----------------------------------------------------------------------
Ran 1 test in 0.053s

OK
test_app_client_buy_valid (test_func.TestFunctionality) ... ok

----------------------------------------------------------------------
Ran 1 test in 0.061s

OK
test_app_client_buy_invalid (test_func.TestFunctionality) ... ok
```



maoqin@maoqin-VirtualBox: ~/lab3/src/test

File Edit View Search Terminal Help

```
----------------------------------------------------------------------
Ran 1 test in 0.054s

OK
test_app_client_buy_outofstock (test_func.TestFunctionality) ... ok

----------------------------------------------------------------------
Ran 1 test in 0.064s

OK
test_microservices_order_cata_valid (test_func.TestFunctionality) ... ok

----------------------------------------------------------------------
Ran 1 test in 0.052s

OK
test_microservices_order_cata_invalid (test_func.TestFunctionality) ... ok

----------------------------------------------------------------------
Ran 1 test in 0.049s

OK
test_app_client_queryOrder_valid (test_func.TestFunctionality) ... ok

----------------------------------------------------------------------
Ran 1 test in 0.048s

OK
test_app_client_queryOrder_invalid (test_func.TestFunctionality) ... ok
```

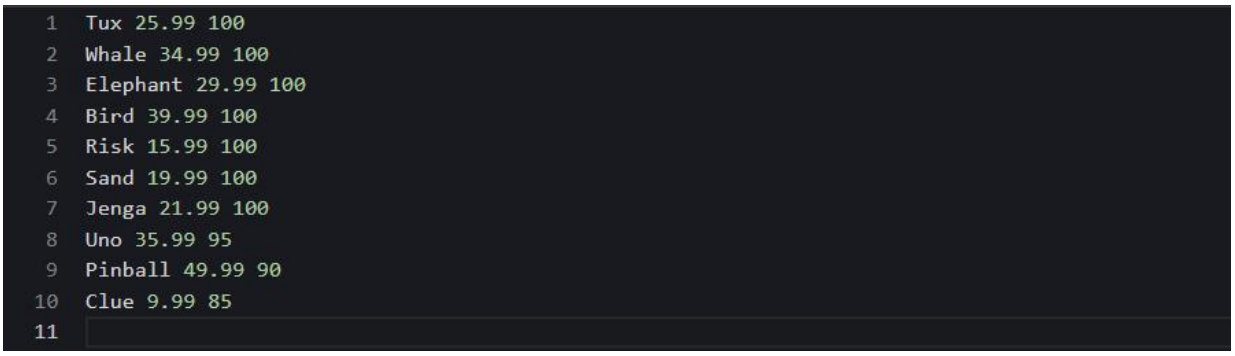


maoqin@maoqin-VirtualBox: ~/lab3/src/test

File Edit View Search Terminal Help

```
----------------------------------------------------------------------
Ran 1 test in 0.046s

OK
test_microservices_frontend_queryOrder_valid (test_func.TestFunctionality) ...
ok

----------------------------------------------------------------------
Ran 1 test in 0.058s

OK
test_microservices_frontend_queryOrder_invalid (test_func.TestFunctionality) ..
. ok

----------------------------------------------------------------------
Ran 1 test in 0.089s

OK
test_microservices_frontend_heartbeat_valid (test_func.TestFunctionality) ... o
k

----------------------------------------------------------------------
Ran 1 test in 0.093s

OK
test_microservices_frontend_notifyNewLeader_valid (test_func.TestFunctionality)
 ... ok

----------------------------------------------------------------------
```

And also, after finishing those 19 test cases, we can see the database at catalog server has been recorded and persisted correctly.



```
Tux 25.99 100
Whale 34.99 100
Elephant 29.99 100
Bird 39.99 100
Risk 15.99 100
Sand 19.99 100
Jenga 21.99 100
Uno 35.99 95
Pinball 49.99 90
Clue 9.99 85
```

Consistency of Database & Order Log Testing

Looking at **"client.py"**, we implemented 3 modes for you.

Mode 1: Query and Buy randomly:

It randomly queries an item, if the returned quantity is greater than 0, with probability "p" (environment variable) it will send an order request.

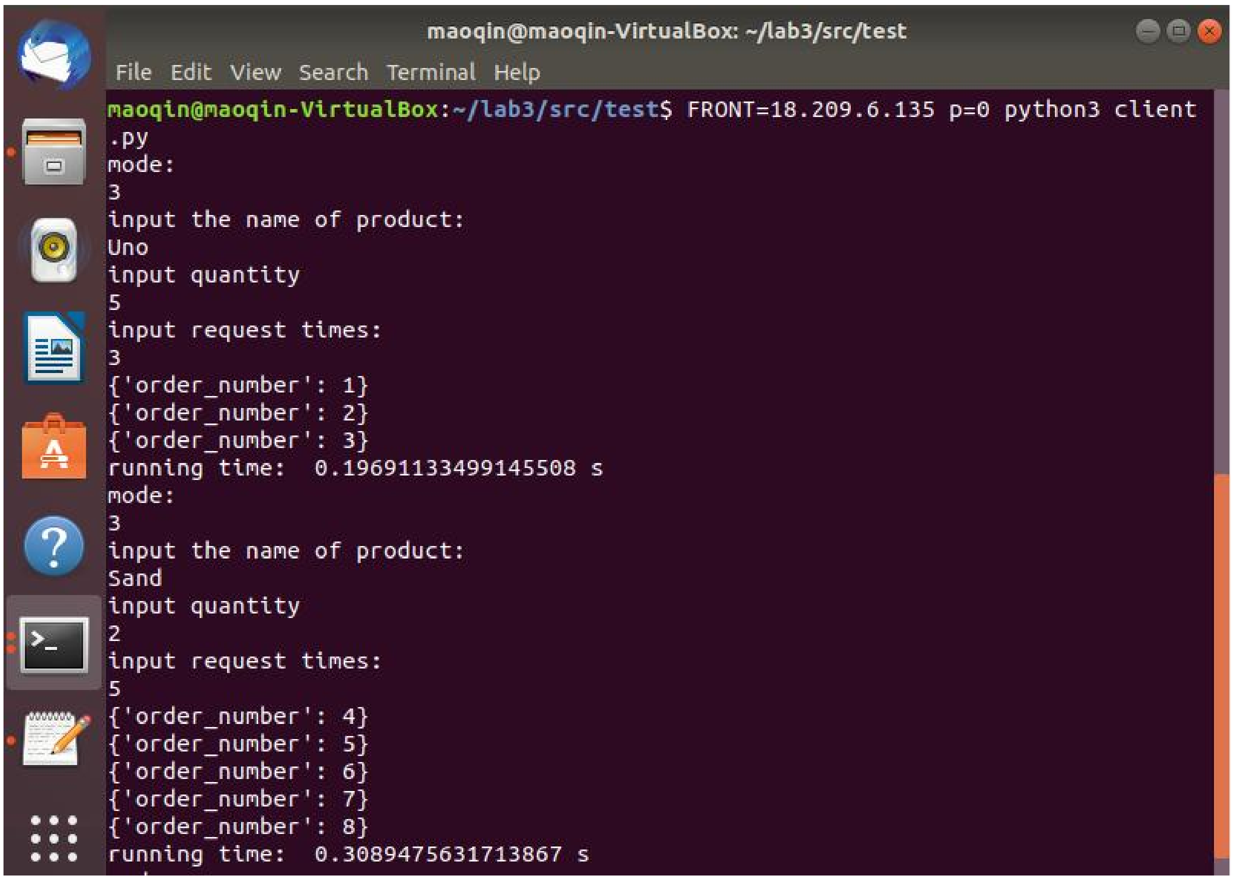
Mode 2: Initiate a serials of Query

You can specify the toy name and query times as you want.

Mode 3: Initiate a serials of Buy

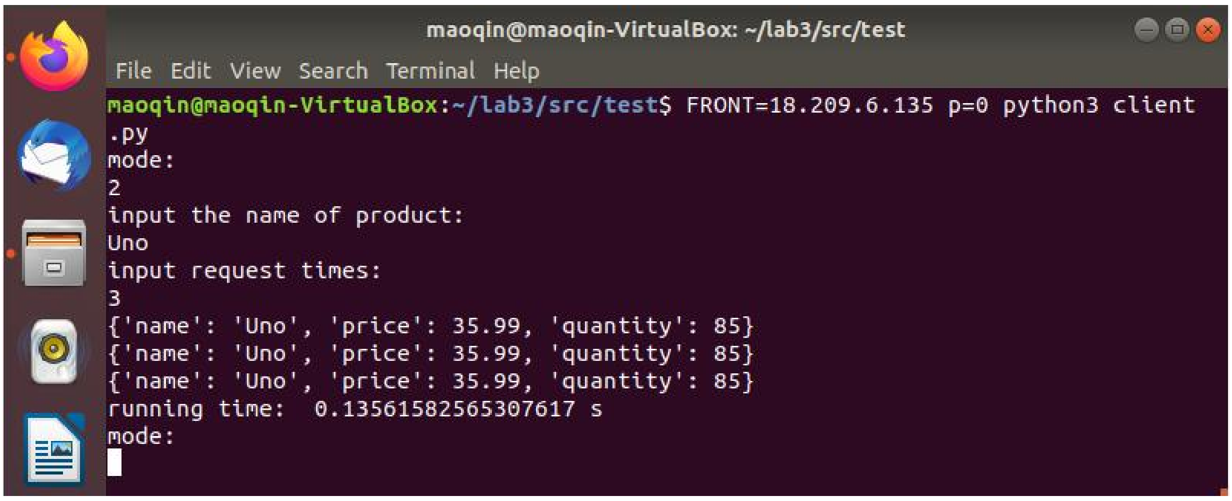
You can specify the toy name, quantity and number of requests as you want.

Test Case: place order totally 8 times (successful)


maoqin@maoqin-VirtualBox: ~/lab3/src/test
File Edit View Search Terminal Help

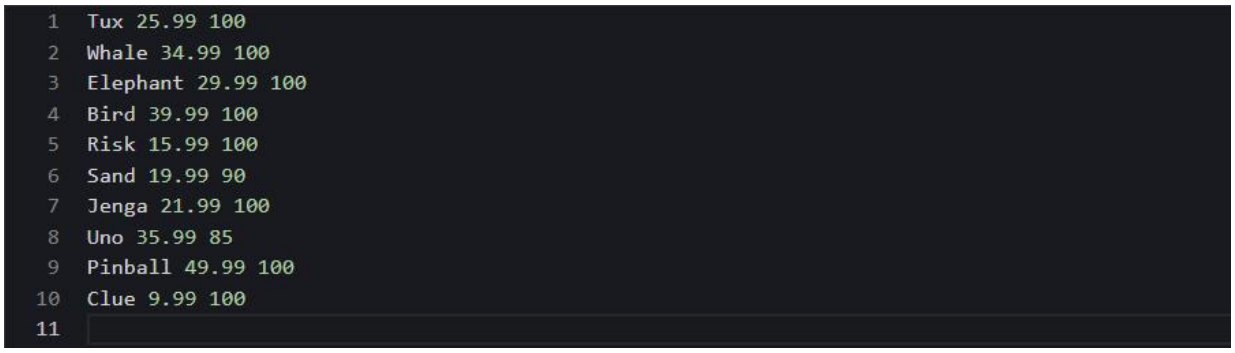
```
maoqin@maoqin-VirtualBox:~/lab3/src/test$ FRONT=18.209.6.135 p=0 python3 client
.py
mode:
3
input the name of product:
Uno
input quantity
5
input request times:
3
{'order_number': 1}
{'order_number': 2}
{'order_number': 3}
running time:  0.19691133499145508 s
mode:
3
input the name of product:
Sand
input quantity
2
input request times:
5
{'order_number': 4}
{'order_number': 5}
{'order_number': 6}
{'order_number': 7}
{'order_number': 8}
running time:  0.3089475631713867 s
```

Query the stock of Uno: the responses are correct


maoqin@maoqin-VirtualBox: ~/lab3/src/test
File Edit View Search Terminal Help

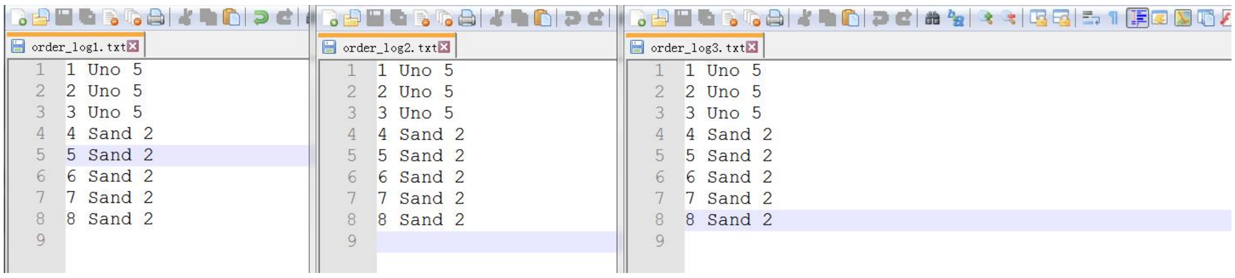
```
maoqin@maoqin-VirtualBox:~/lab3/src/test$ FRONT=18.209.6.135 p=0 python3 client
.py
mode:
2
input the name of product:
Uno
input request times:
3
{'name': 'Uno', 'price': 35.99, 'quantity': 85}
{'name': 'Uno', 'price': 35.99, 'quantity': 85}
{'name': 'Uno', 'price': 35.99, 'quantity': 85}
running time:  0.13561582565307617 s
mode:
```

Checkout the database at catalog server: the stock are correct



```
Tux 25.99 100
Whale 34.99 100
Elephant 29.99 100
Bird 39.99 100
Risk 15.99 100
Sand 19.99 90
Jenga 21.99 100
Uno 35.99 85
Pinball 49.99 100
Clue 9.99 100
```

Checkout the order log at each order server: all of the three order logs are consistent and correct



```
1 Uno 5
2 Uno 5
3 Uno 5
4 Sand 2
5 Sand 2
6 Sand 2
7 Sand 2
8 Sand 2
```

```
1 Uno 5
2 Uno 5
3 Uno 5
4 Sand 2
5 Sand 2
6 Sand 2
7 Sand 2
8 Sand 2
```

```
1 Uno 5
2 Uno 5
3 Uno 5
4 Sand 2
5 Sand 2
6 Sand 2
7 Sand 2
8 Sand 2
```

3. Load Test Output

Concurrent Requests- Looking at **"test_load.py"**, it automatically sends **1000** Query, Buy or queryOrder requests. Python unittest can help measure the total latency seen by clients in this case. Hence, in terms of average latency for each request, we should divide the total time by **1000**.

For different type of requests, we repeatedly run 5 clients at the same time, and measure the total latency seen by each client. There are 3 commands for each performance testing.

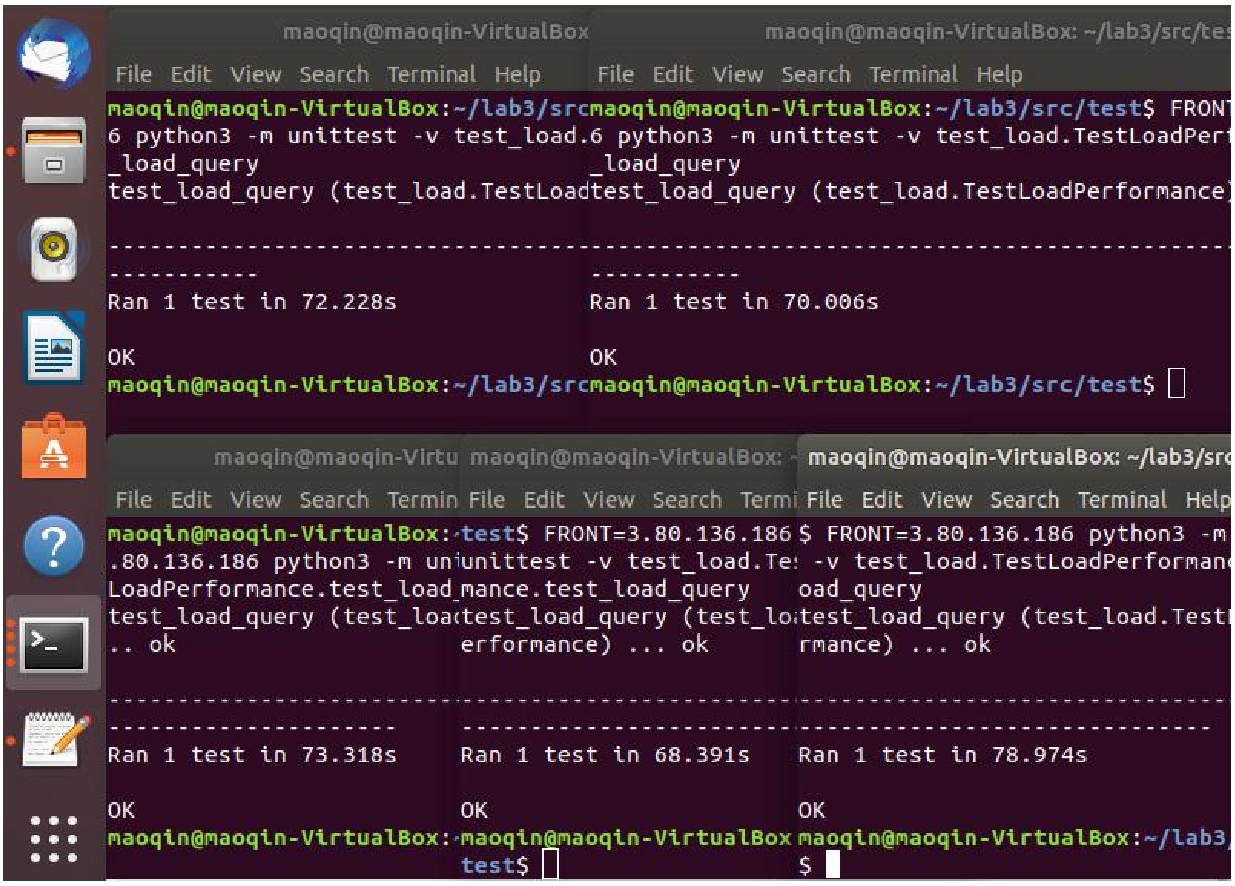
$ FRONT=<IP address> python3 -m unittest -v test_load.TestLoadPerformance.test_load_query

$ FRONT=<IP address> python3 -m unittest -v test_load.TestLoadPerformance.test_load_buy

$ FRONT=<IP address> python3 -m unittest -v test_load.TestLoadPerformance.test_load_queryOrder

We have analyzed the average latency for different requests in "evaluation" document. Please checkout the details there.

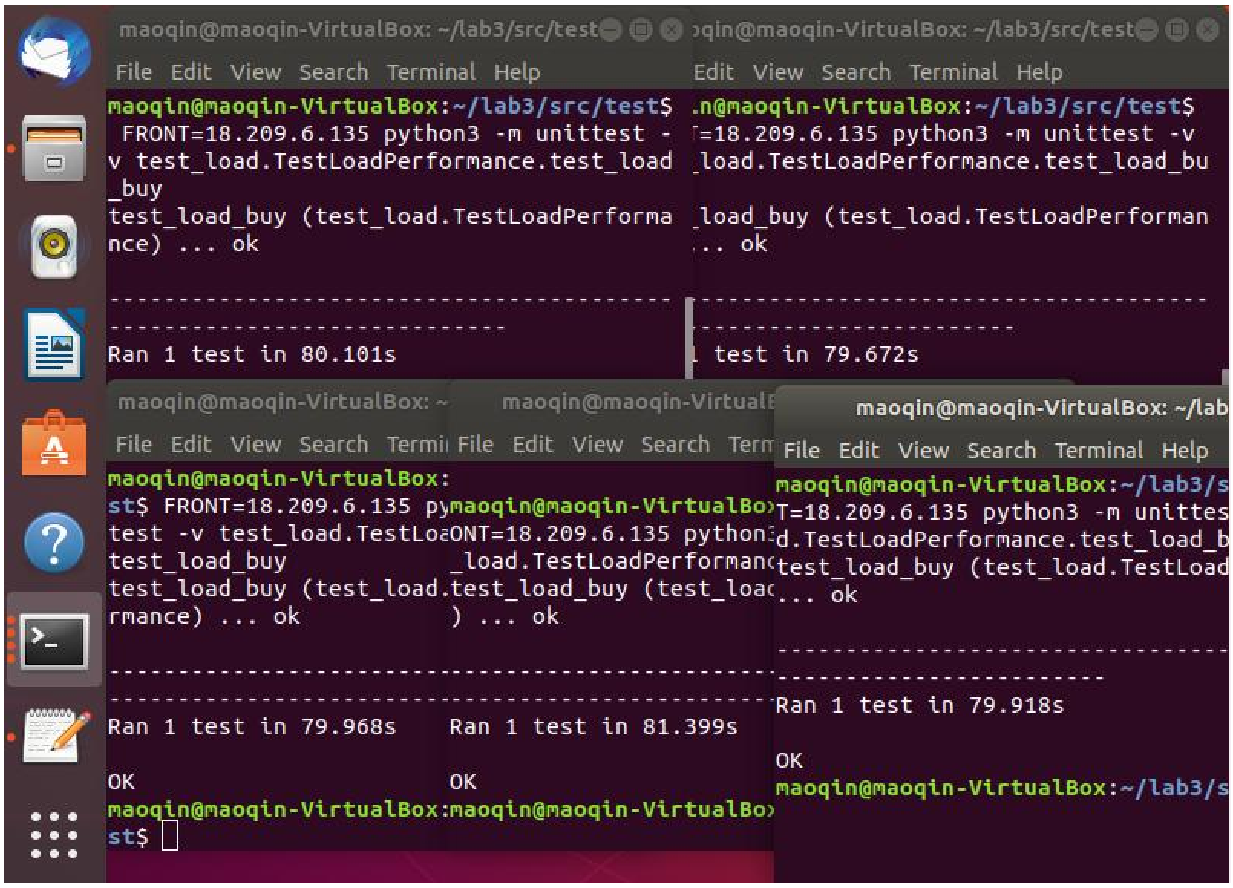
Query Load Test: shows total latency of 1000 Query calls



```
maoqin@maoqin-VirtualBox:~/lab3/srcmaoqin@maoqin-VirtualBox:~/lab3/src/test$ FRONT
6 python3 -m unittest -v test_load.6 python3 -m unittest -v test_load.TestLoadPerf
_load_query                        _load_query
test_load_query (test_load.TestLoadtest_load_query (test_load.TestLoadPerformance)

----------------------------------------------------------------------------------
-----------                        -----------
Ran 1 test in 72.228s              Ran 1 test in 70.006s

OK                                 OK
maoqin@maoqin-VirtualBox:~/lab3/srcmaoqin@maoqin-VirtualBox:~/lab3/src/test$
```

```
maoqin@maoqin-VirtualBox:~test$ FRONT=3.80.136.186 $ FRONT=3.80.136.186 python3 -m
.80.136.186 python3 -m uniunittest -v test_load.Te: -v test_load.TestLoadPerforman
LoadPerformance.test_load_mance.test_load_query    oad_query
test_load_query (test_loatest_load_query (test_lo:test_load_query (test_load.Test
.. ok                     erformance) ... ok       rmance) ... ok

---------------------------------------------------------------------------------
---------------------                              ------------------------------
Ran 1 test in 73.318s     Ran 1 test in 68.391s    Ran 1 test in 78.974s

OK                        OK                       OK
maoqin@maoqin-VirtualBox:~maoqin@maoqin-VirtualBox maoqin@maoqin-VirtualBox:~/lab3
                          test$                    $
```

Buy Load Test: shows total latency of 1000 Buy calls

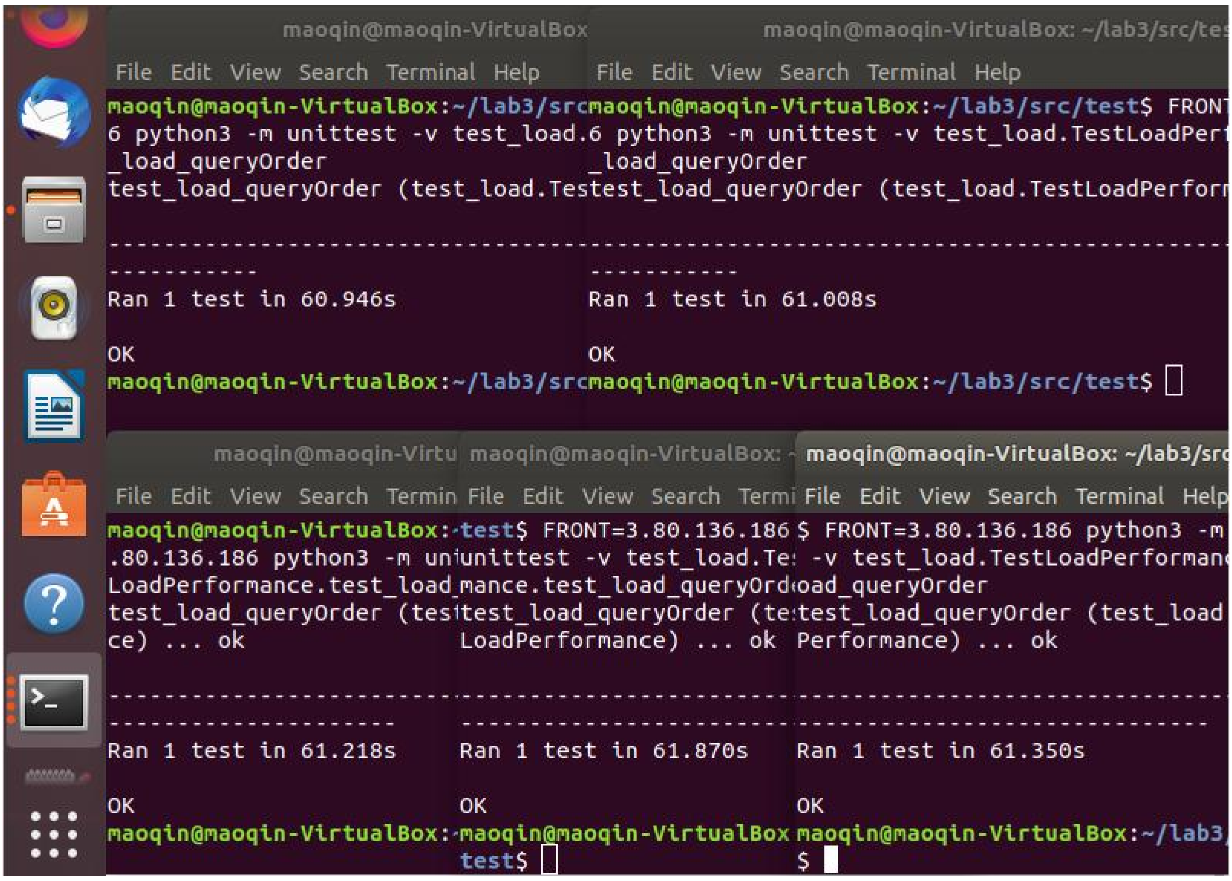

maoqin@maoqin-VirtualBox: ~/lab3/src/test
File Edit View Search Terminal Help
ɔqin@maoqin-VirtualBox: ~/lab3/src/test
Edit View Search Terminal Help

```
maoqin@maoqin-VirtualBox:~/lab3/src/test$  .n@maoqin-VirtualBox:~/lab3/src/test$
 FRONT=18.209.6.135 python3 -m unittest -  T=18.209.6.135 python3 -m unittest -v
v test_load.TestLoadPerformance.test_load   load.TestLoadPerformance.test_load_bu
_buy
test_load_buy (test_load.TestLoadPerforma   load_buy (test_load.TestLoadPerforman
nce) ... ok                                 .. ok

-----------------------------------------  ----------------------------------------
----------------------------                -----------------------
Ran 1 test in 80.101s                       test in 79.672s
```

maoqin@maoqin-VirtualBox: ~
File Edit View Search Termi
maoqin@maoqin-VirtualE
File Edit View Search Term
maoqin@maoqin-VirtualBox: ~/lab
File Edit View Search Terminal Help

```
maoqin@maoqin-VirtualBox:                                    maoqin@maoqin-VirtualBox:~/lab3/s
st$ FRONT=18.209.6.135 pymaoqin@maoqin-VirtualBox T=18.209.6.135 python3 -m unittes
test -v test_load.TestLoaONT=18.209.6.135 python: d.TestLoadPerformance.test_load_b
test_load_buy             _load.TestLoadPerformanc test_load_buy (test_load.TestLoad
test_load_buy (test_load.test_load_buy (test_loac ... ok
rmance) ... ok            ) ... ok
                                                   ----------------------------------
-------------------------------------------------- -----------------------
---------------------------------------------------Ran 1 test in 79.918s
Ran 1 test in 79.968s     Ran 1 test in 81.399s
                                                   OK
OK                        OK                       maoqin@maoqin-VirtualBox:~/lab3/s
maoqin@maoqin-VirtualBox:maoqin@maoqin-VirtualBox
st$ 
```

queryOrder Load Test: shows total latency of 1000 queryOrder calls


maoqin@maoqin-VirtualBox
File Edit View Search Terminal Help
maoqin@maoqin-VirtualBox: ~/lab3/src/tes
File Edit View Search Terminal Help

```
maoqin@maoqin-VirtualBox:~/lab3/srcmaoqin@maoqin-VirtualBox:~/lab3/src/test$ FRONT
6 python3 -m unittest -v test_load.6 python3 -m unittest -v test_load.TestLoadPerf
_load_queryOrder                   _load_queryOrder
test_load_queryOrder (test_load.Testest_load_queryOrder (test_load.TestLoadPerfor

-----------------------------------------------------------------------------------
-----------                        -----------
Ran 1 test in 60.946s              Ran 1 test in 61.008s

OK                                 OK
maoqin@maoqin-VirtualBox:~/lab3/srcmaoqin@maoqin-VirtualBox:~/lab3/src/test$ 
```

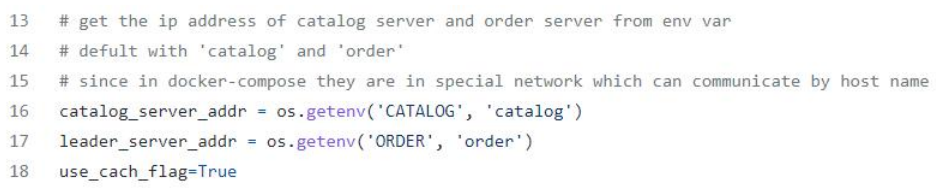
maoqin@maoqin-Virtu
File Edit View Search Termin
maoqin@maoqin-VirtualBox: ~
File Edit View Search Termi
maoqin@maoqin-VirtualBox: ~/lab3/src
File Edit View Search Terminal Help

```
maoqin@maoqin-VirtualBox:-test$ FRONT=3.80.136.186 $ FRONT=3.80.136.186 python3 -m
.80.136.186 python3 -m uniunittest -v test_load.Te: -v test_load.TestLoadPerforman
LoadPerformance.test_load mance.test_load_queryOrdoad_queryOrder
test_load_queryOrder (tesitest_load_queryOrder (te:test_load_queryOrder (test_load
ce) ... ok                LoadPerformance) ... ok  Performance) ... ok

---------------------------------------------------------------------------------
--------------------      -----------------------  ---------------------------------
Ran 1 test in 61.218s     Ran 1 test in 61.870s    Ran 1 test in 61.350s

OK                        OK                       OK
maoqin@maoqin-VirtualBox:-maoqin@maoqin-VirtualBox maoqin@maoqin-VirtualBox:~/lab3/
                          test$                    $ 
```

4. Caching Test Output

In order to estimate how much benefits does caching. We are firstly measuring the latency seen by each client for different type requests with caching turned on. Change the probability p of a follow up buy request from 0 to 80%, with an increment of 20%, and record the result for each p setting. And then do the same experiments but with caching turned off.

Caching Switch: look at "front_end.py", you can switch the state of caching by modifying the global variable "**use_cach_flag**".



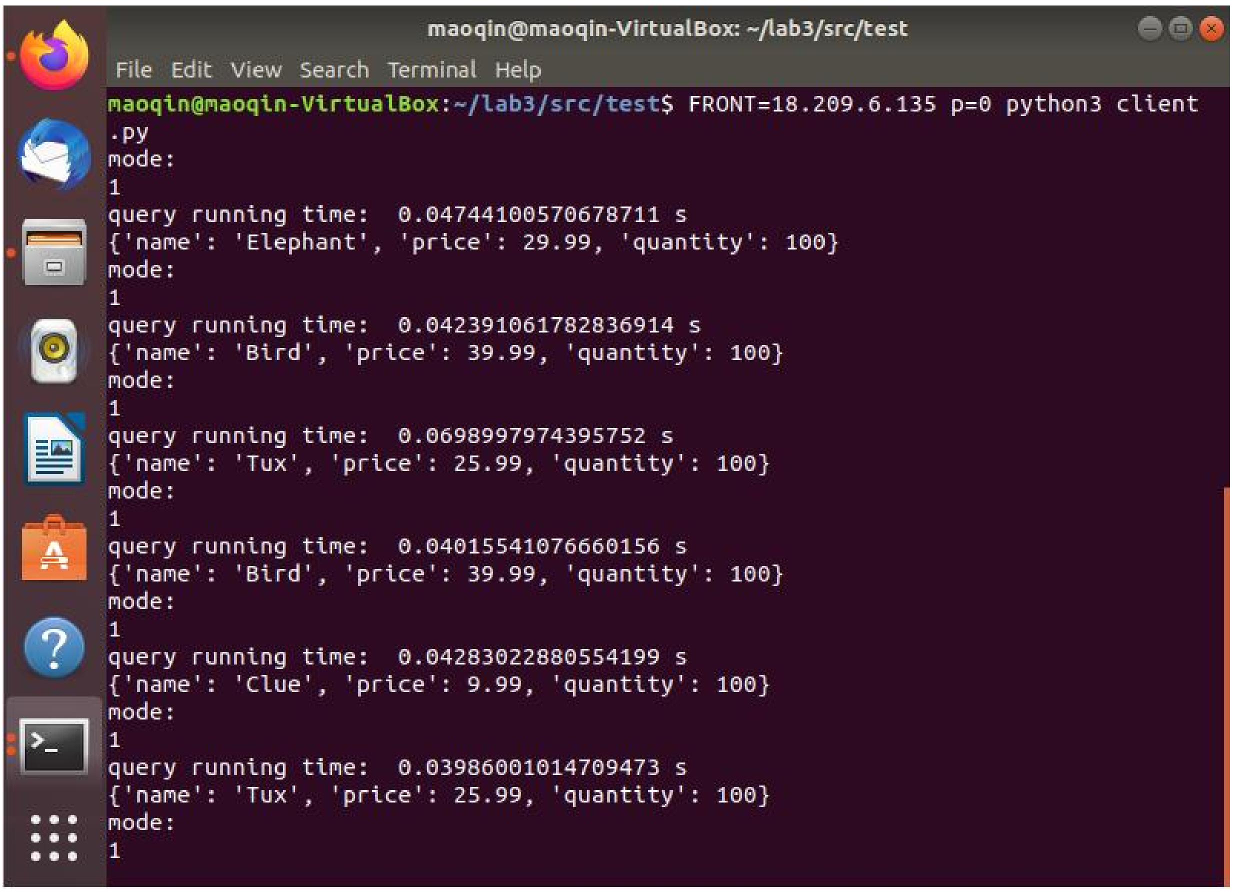
```python
# get the ip address of catalog server and order server from env var
# defult with 'catalog' and 'order'
# since in docker-compose they are in special network which can communicate by host name
catalog_server_addr = os.getenv('CATALOG', 'catalog')
leader_server_addr = os.getenv('ORDER', 'order')
use_cach_flag=True
```

The command you type on your local machine:

$ FRONT=<IP address> p=<probability> python3 client.py

For each experiment with different p, we are testing **multiple times**, and record the **average latency**. The output screenshots for different p and caching state are shown as follows:

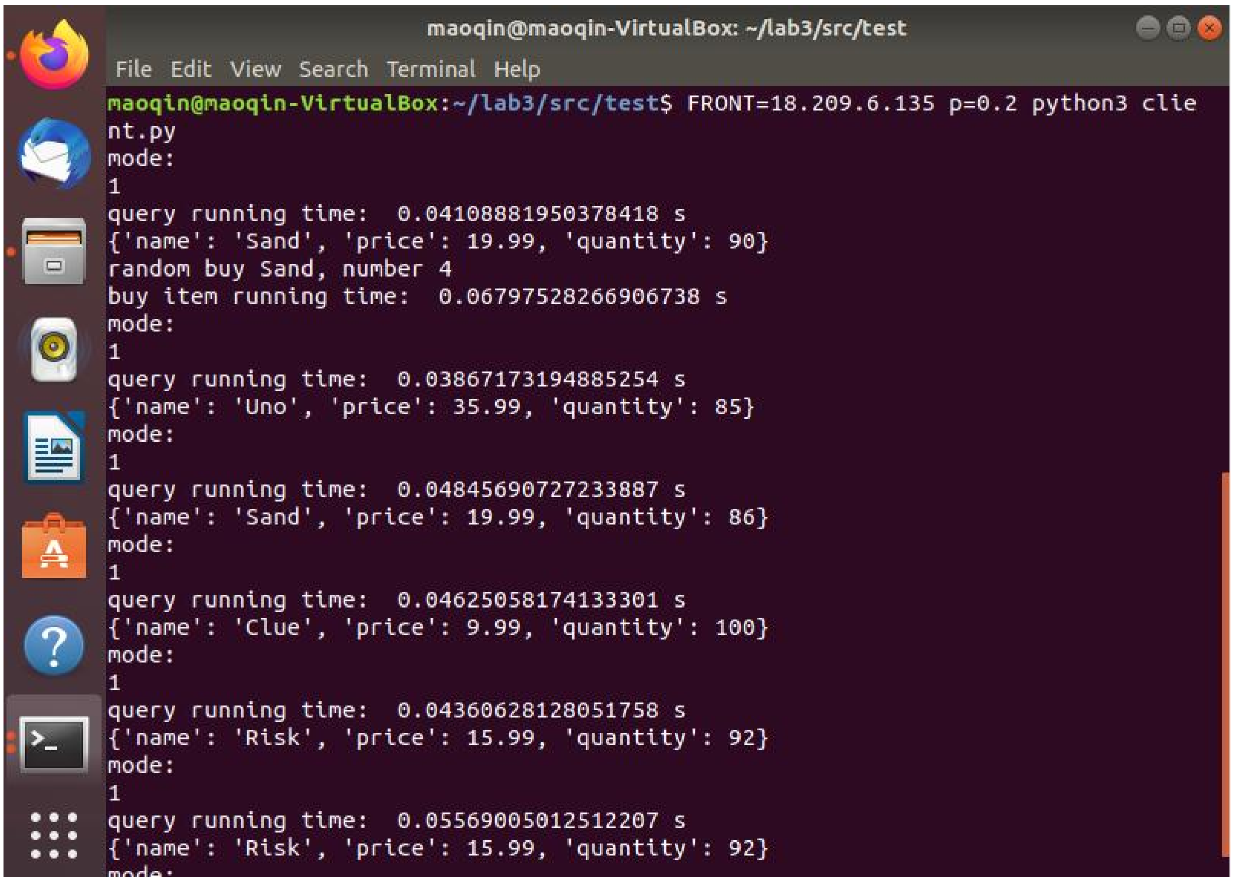
Latency of p=0 with caching turned on:



maoqin@maoqin-VirtualBox: ~/lab3/src/test
File Edit View Search Terminal Help

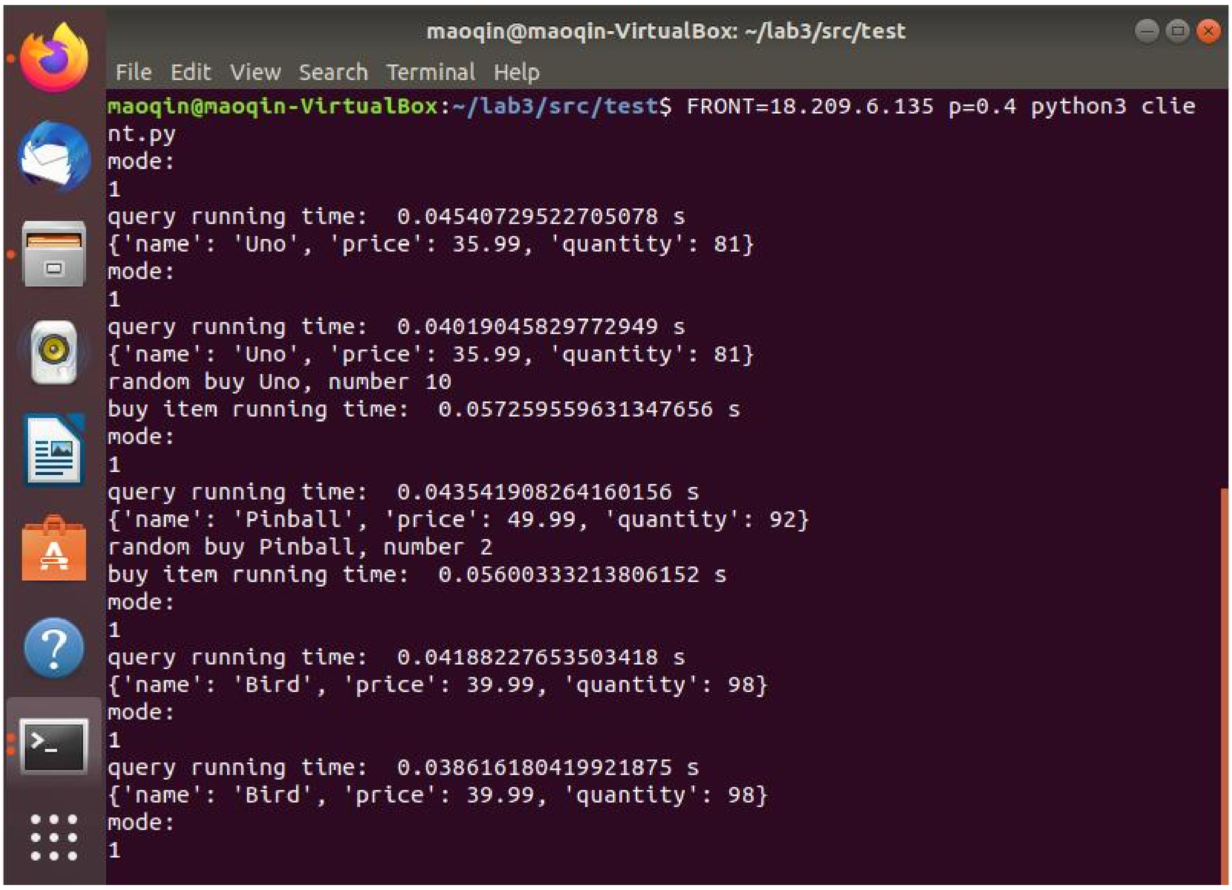
```
maoqin@maoqin-VirtualBox:~/lab3/src/test$ FRONT=18.209.6.135 p=0 python3 client
.py
mode:
1
query running time:  0.04744100570678711 s
{'name': 'Elephant', 'price': 29.99, 'quantity': 100}
mode:
1
query running time:  0.042391061782836914 s
{'name': 'Bird', 'price': 39.99, 'quantity': 100}
mode:
1
query running time:  0.0698997974395752 s
{'name': 'Tux', 'price': 25.99, 'quantity': 100}
mode:
1
query running time:  0.04015541076660156 s
{'name': 'Bird', 'price': 39.99, 'quantity': 100}
mode:
1
query running time:  0.04283022880554199 s
{'name': 'Clue', 'price': 9.99, 'quantity': 100}
mode:
1
query running time:  0.03986001014709473 s
{'name': 'Tux', 'price': 25.99, 'quantity': 100}
mode:
1
```

Latency of p=0.2 with caching turned on:


maoqin@maoqin-VirtualBox: ~/lab3/src/test
File Edit View Search Terminal Help

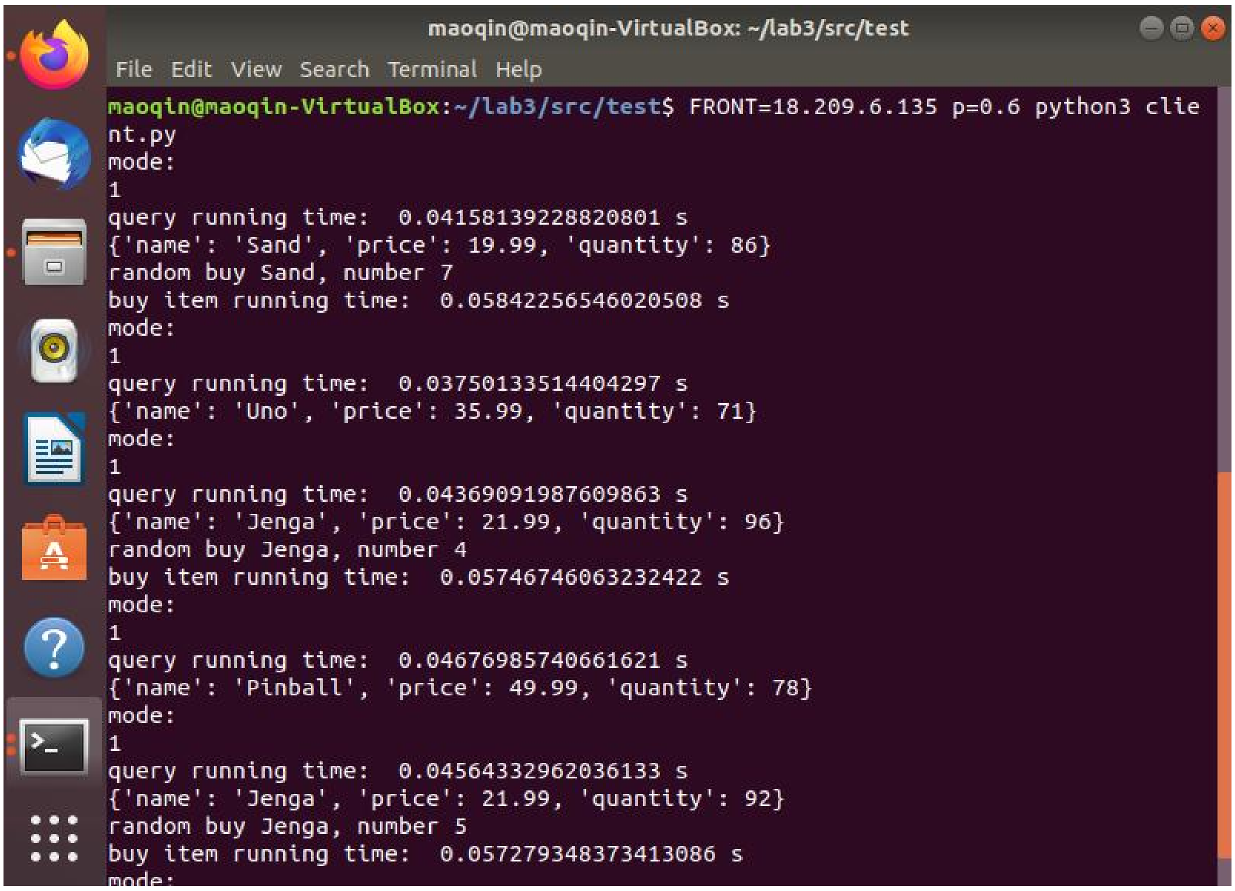
```
maoqin@maoqin-VirtualBox:~/lab3/src/test$ FRONT=18.209.6.135 p=0.2 python3 clie
nt.py
mode:
1
query running time:  0.04108881950378418 s
{'name': 'Sand', 'price': 19.99, 'quantity': 90}
random buy Sand, number 4
buy item running time:  0.06797528266906738 s
mode:
1
query running time:  0.03867173194885254 s
{'name': 'Uno', 'price': 35.99, 'quantity': 85}
mode:
1
query running time:  0.04845690727233887 s
{'name': 'Sand', 'price': 19.99, 'quantity': 86}
mode:
1
query running time:  0.04625058174133301 s
{'name': 'Clue', 'price': 9.99, 'quantity': 100}
mode:
1
query running time:  0.04360628128051758 s
{'name': 'Risk', 'price': 15.99, 'quantity': 92}
mode:
1
query running time:  0.05569005012512207 s
{'name': 'Risk', 'price': 15.99, 'quantity': 92}
mode:
```

Latency of p=0.4 with caching turned on:


maoqin@maoqin-VirtualBox: ~/lab3/src/test
File Edit View Search Terminal Help

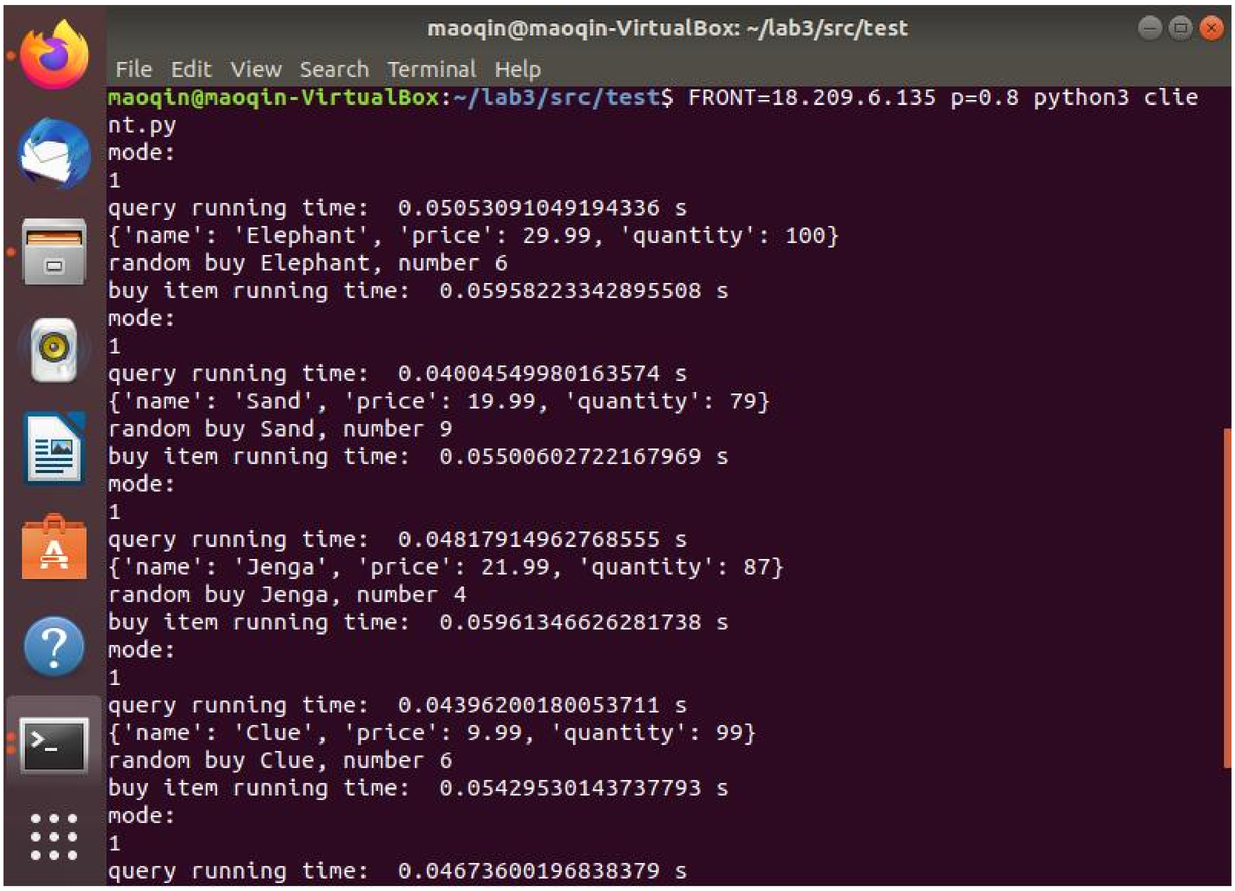
```
maoqin@maoqin-VirtualBox:~/lab3/src/test$ FRONT=18.209.6.135 p=0.4 python3 clie
nt.py
mode:
1
query running time:  0.04540729522705078 s
{'name': 'Uno', 'price': 35.99, 'quantity': 81}
mode:
1
query running time:  0.04019045829772949 s
{'name': 'Uno', 'price': 35.99, 'quantity': 81}
random buy Uno, number 10
buy item running time:  0.057259559631347656 s
mode:
1
query running time:  0.043541908264160156 s
{'name': 'Pinball', 'price': 49.99, 'quantity': 92}
random buy Pinball, number 2
buy item running time:  0.05600333213806152 s
mode:
1
query running time:  0.04188227653503418 s
{'name': 'Bird', 'price': 39.99, 'quantity': 98}
mode:
1
query running time:  0.038616180419921875 s
{'name': 'Bird', 'price': 39.99, 'quantity': 98}
mode:
1
```

Latency of p=0.6 with caching turned on:


maoqin@maoqin-VirtualBox: ~/lab3/src/test
File Edit View Search Terminal Help

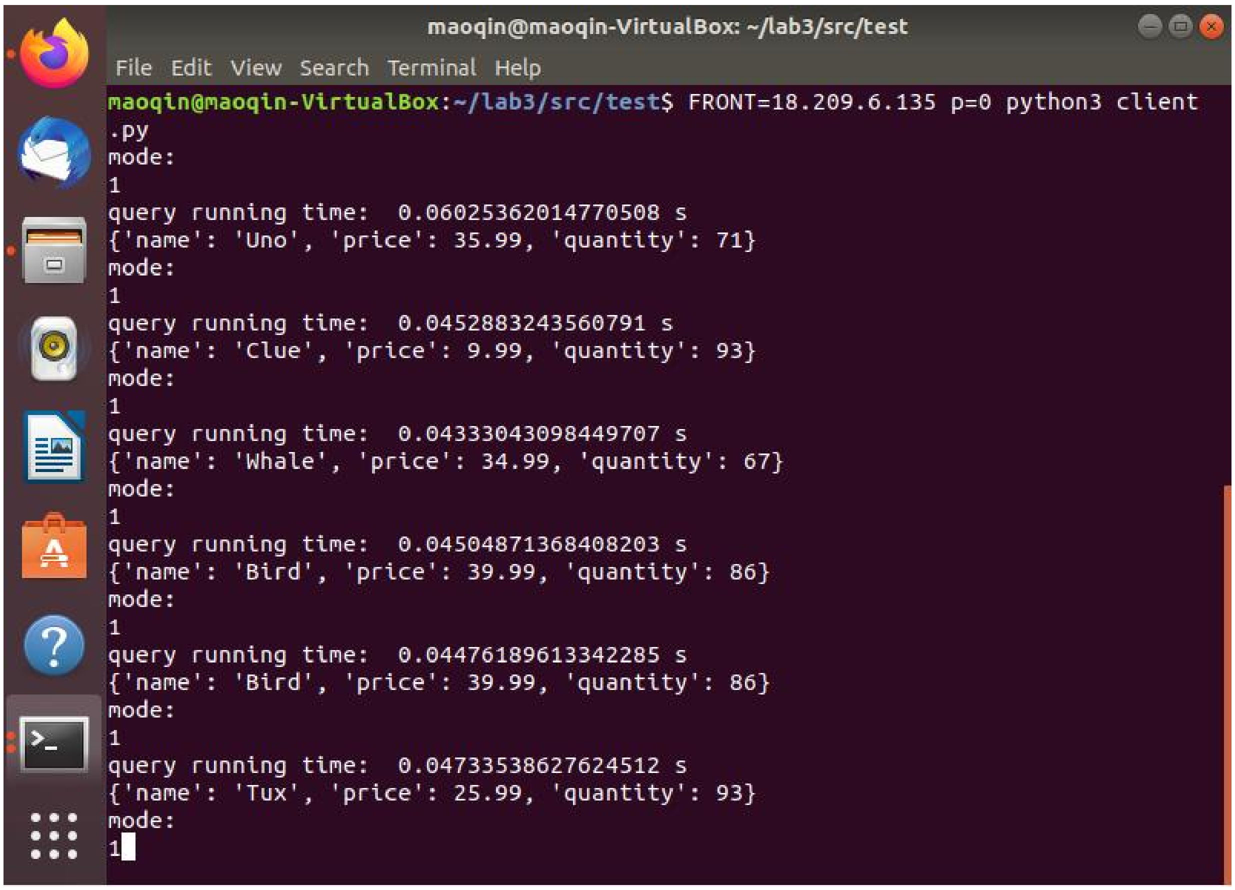
```
maoqin@maoqin-VirtualBox:~/lab3/src/test$ FRONT=18.209.6.135 p=0.6 python3 clie
nt.py
mode:
1
query running time:  0.04158139228820801 s
{'name': 'Sand', 'price': 19.99, 'quantity': 86}
random buy Sand, number 7
buy item running time:  0.05842256546020508 s
mode:
1
query running time:  0.03750133514404297 s
{'name': 'Uno', 'price': 35.99, 'quantity': 71}
mode:
1
query running time:  0.04369091987609863 s
{'name': 'Jenga', 'price': 21.99, 'quantity': 96}
random buy Jenga, number 4
buy item running time:  0.05746746063232422 s
mode:
1
query running time:  0.04676985740661621 s
{'name': 'Pinball', 'price': 49.99, 'quantity': 78}
mode:
1
query running time:  0.04564332962036133 s
{'name': 'Jenga', 'price': 21.99, 'quantity': 92}
random buy Jenga, number 5
buy item running time:  0.05727934837341308б s
mode:
```

Latency of p=0.8 with caching turned on:


maoqin@maoqin-VirtualBox: ~/lab3/src/test
File Edit View Search Terminal Help

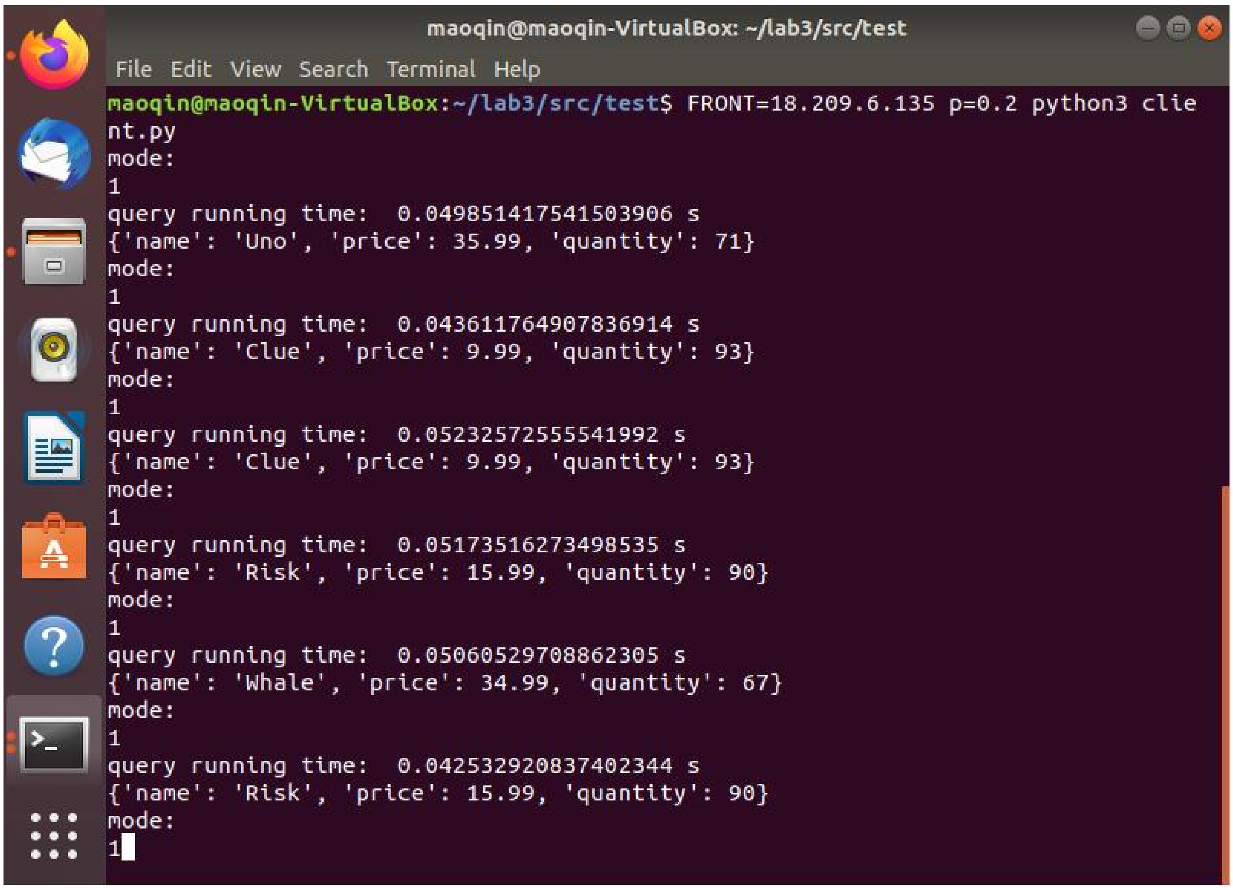
```
maoqin@maoqin-VirtualBox:~/lab3/src/test$ FRONT=18.209.6.135 p=0.8 python3 clie
nt.py
mode:
1
query running time:  0.05053091049194336 s
{'name': 'Elephant', 'price': 29.99, 'quantity': 100}
random buy Elephant, number 6
buy item running time:  0.05958223342895508 s
mode:
1
query running time:  0.04004549980163574 s
{'name': 'Sand', 'price': 19.99, 'quantity': 79}
random buy Sand, number 9
buy item running time:  0.05500602722167969 s
mode:
1
query running time:  0.04817914962768555 s
{'name': 'Jenga', 'price': 21.99, 'quantity': 87}
random buy Jenga, number 4
buy item running time:  0.05961346626281738 s
mode:
1
query running time:  0.04396200180053711 s
{'name': 'Clue', 'price': 9.99, 'quantity': 99}
random buy Clue, number 6
buy item running time:  0.05429530143737793 s
mode:
1
query running time:  0.04673600196838379 s
```

Latency of p=0 with caching turned off:


maoqin@maoqin-VirtualBox: ~/lab3/src/test
File Edit View Search Terminal Help

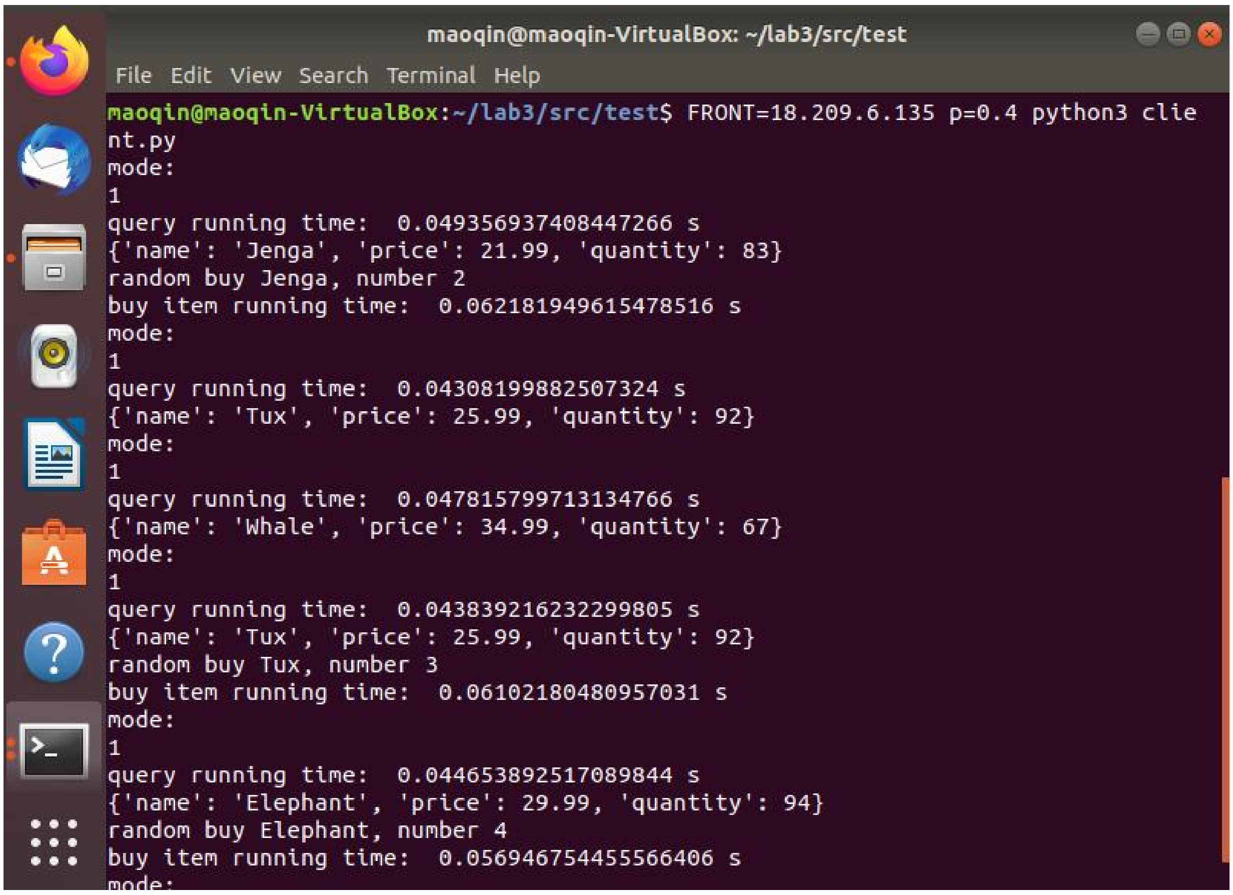
```
maoqin@maoqin-VirtualBox:~/lab3/src/test$ FRONT=18.209.6.135 p=0 python3 client
.py
mode:
1
query running time:  0.06025362014770508 s
{'name': 'Uno', 'price': 35.99, 'quantity': 71}
mode:
1
query running time:  0.0452883243560791 s
{'name': 'Clue', 'price': 9.99, 'quantity': 93}
mode:
1
query running time:  0.04333043098449707 s
{'name': 'Whale', 'price': 34.99, 'quantity': 67}
mode:
1
query running time:  0.04504871368408203 s
{'name': 'Bird', 'price': 39.99, 'quantity': 86}
mode:
1
query running time:  0.04476189613342285 s
{'name': 'Bird', 'price': 39.99, 'quantity': 86}
mode:
1
query running time:  0.04733538627624512 s
{'name': 'Tux', 'price': 25.99, 'quantity': 93}
mode:
1
```

Latency of p=0.2 with caching turned off:


maoqin@maoqin-VirtualBox: ~/lab3/src/test
File Edit View Search Terminal Help

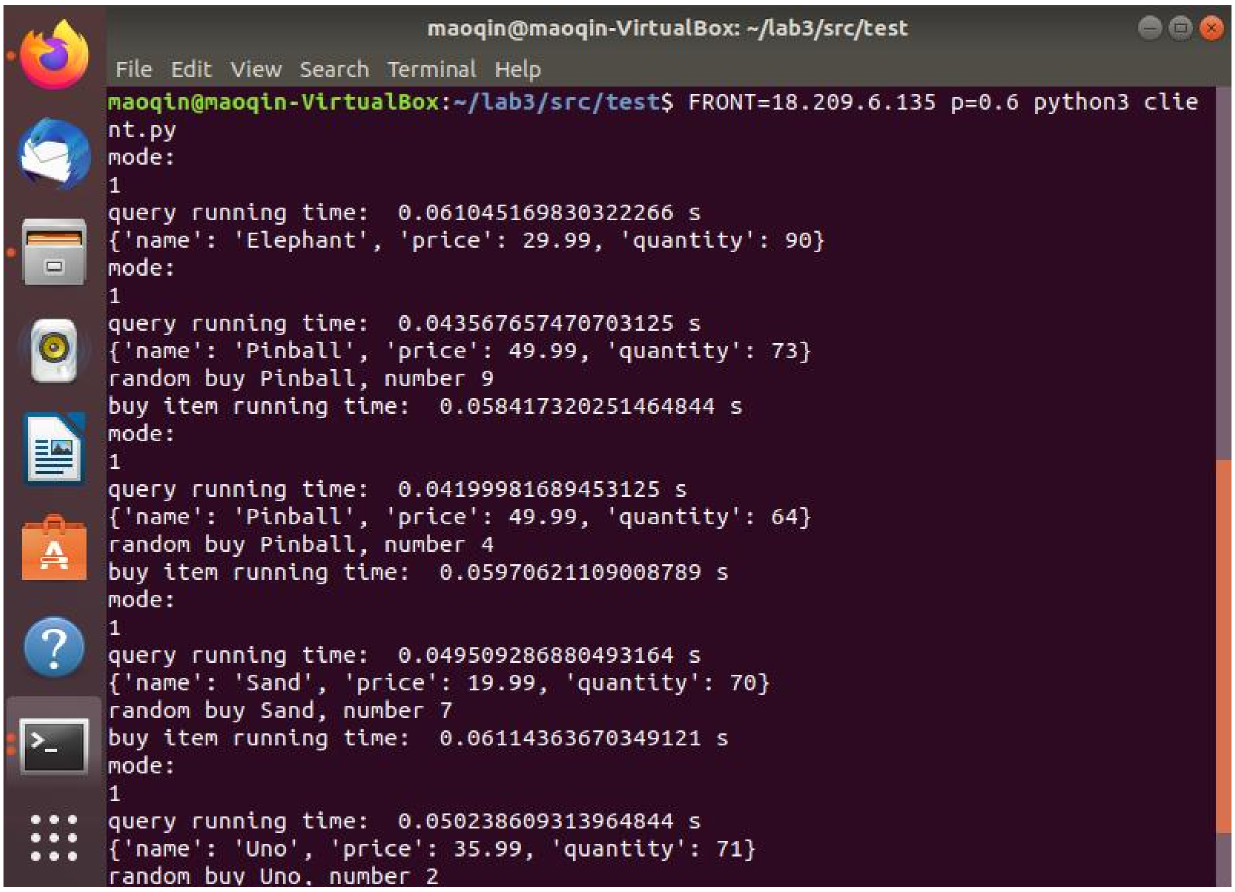
```
maoqin@maoqin-VirtualBox:~/lab3/src/test$ FRONT=18.209.6.135 p=0.2 python3 clie
nt.py
mode:
1
query running time:  0.049851417541503906 s
{'name': 'Uno', 'price': 35.99, 'quantity': 71}
mode:
1
query running time:  0.043611764907836914 s
{'name': 'Clue', 'price': 9.99, 'quantity': 93}
mode:
1
query running time:  0.05232572555541992 s
{'name': 'Clue', 'price': 9.99, 'quantity': 93}
mode:
1
query running time:  0.05173516273498535 s
{'name': 'Risk', 'price': 15.99, 'quantity': 90}
mode:
1
query running time:  0.05060529708862305 s
{'name': 'Whale', 'price': 34.99, 'quantity': 67}
mode:
1
query running time:  0.042532920837402344 s
{'name': 'Risk', 'price': 15.99, 'quantity': 90}
mode:
1
```

Latency of p=0.4 with caching turned off:



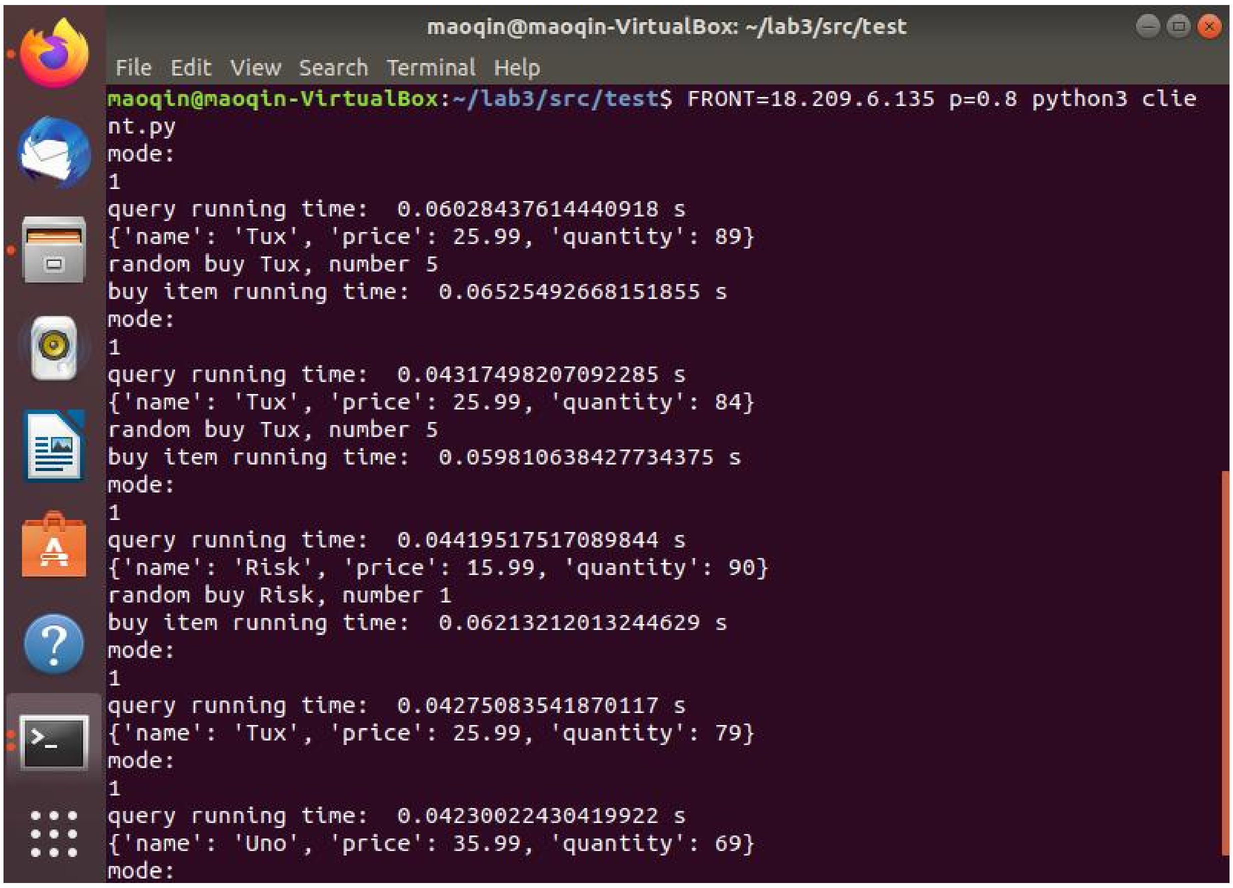
```
maoqin@maoqin-VirtualBox:~/lab3/src/test$ FRONT=18.209.6.135 p=0.4 python3 clie
nt.py
mode:
1
query running time:  0.049356937408447266 s
{'name': 'Jenga', 'price': 21.99, 'quantity': 83}
random buy Jenga, number 2
buy item running time:  0.062181949615478516 s
mode:
1
query running time:  0.04308199882507324 s
{'name': 'Tux', 'price': 25.99, 'quantity': 92}
mode:
1
query running time:  0.047815799713134766 s
{'name': 'Whale', 'price': 34.99, 'quantity': 67}
mode:
1
query running time:  0.043839216232299805 s
{'name': 'Tux', 'price': 25.99, 'quantity': 92}
random buy Tux, number 3
buy item running time:  0.06102180480957031 s
mode:
1
query running time:  0.044653892517089844 s
{'name': 'Elephant', 'price': 29.99, 'quantity': 94}
random buy Elephant, number 4
buy item running time:  0.056946754455566406 s
mode:
```

Latency of p=0.6 with caching turned off:



```
maoqin@maoqin-VirtualBox:~/lab3/src/test$ FRONT=18.209.6.135 p=0.6 python3 clie
nt.py
mode:
1
query running time:  0.061045169830322266 s
{'name': 'Elephant', 'price': 29.99, 'quantity': 90}
mode:
1
query running time:  0.043567657470703125 s
{'name': 'Pinball', 'price': 49.99, 'quantity': 73}
random buy Pinball, number 9
buy item running time:  0.058417320251464844 s
mode:
1
query running time:  0.04199981689453125 s
{'name': 'Pinball', 'price': 49.99, 'quantity': 64}
random buy Pinball, number 4
buy item running time:  0.05970621109008789 s
mode:
1
query running time:  0.049509286880493164 s
{'name': 'Sand', 'price': 19.99, 'quantity': 70}
random buy Sand, number 7
buy item running time:  0.06114363670349121 s
mode:
1
query running time:  0.050238609313964844 s
{'name': 'Uno', 'price': 35.99, 'quantity': 71}
random buy Uno, number 2
```

Latency of p=0.8 with caching turned off:


maoqin@maoqin-VirtualBox: ~/lab3/src/test
File Edit View Search Terminal Help

```
maoqin@maoqin-VirtualBox:~/lab3/src/test$ FRONT=18.209.6.135 p=0.8 python3 clie
nt.py
mode:
1
query running time:  0.06028437614440918 s
{'name': 'Tux', 'price': 25.99, 'quantity': 89}
random buy Tux, number 5
buy item running time:  0.06525492668151855 s
mode:
1
query running time:  0.04317498207092285 s
{'name': 'Tux', 'price': 25.99, 'quantity': 84}
random buy Tux, number 5
buy item running time:  0.0598106384277734375 s
mode:
1
query running time:  0.04419517517089844 s
{'name': 'Risk', 'price': 15.99, 'quantity': 90}
random buy Risk, number 1
buy item running time:  0.06213212013244629 s
mode:
1
query running time:  0.04275083541870117 s
{'name': 'Tux', 'price': 25.99, 'quantity': 79}
mode:
1
query running time:  0.04230022430419922 s
{'name': 'Uno', 'price': 35.99, 'quantity': 69}
mode:
```

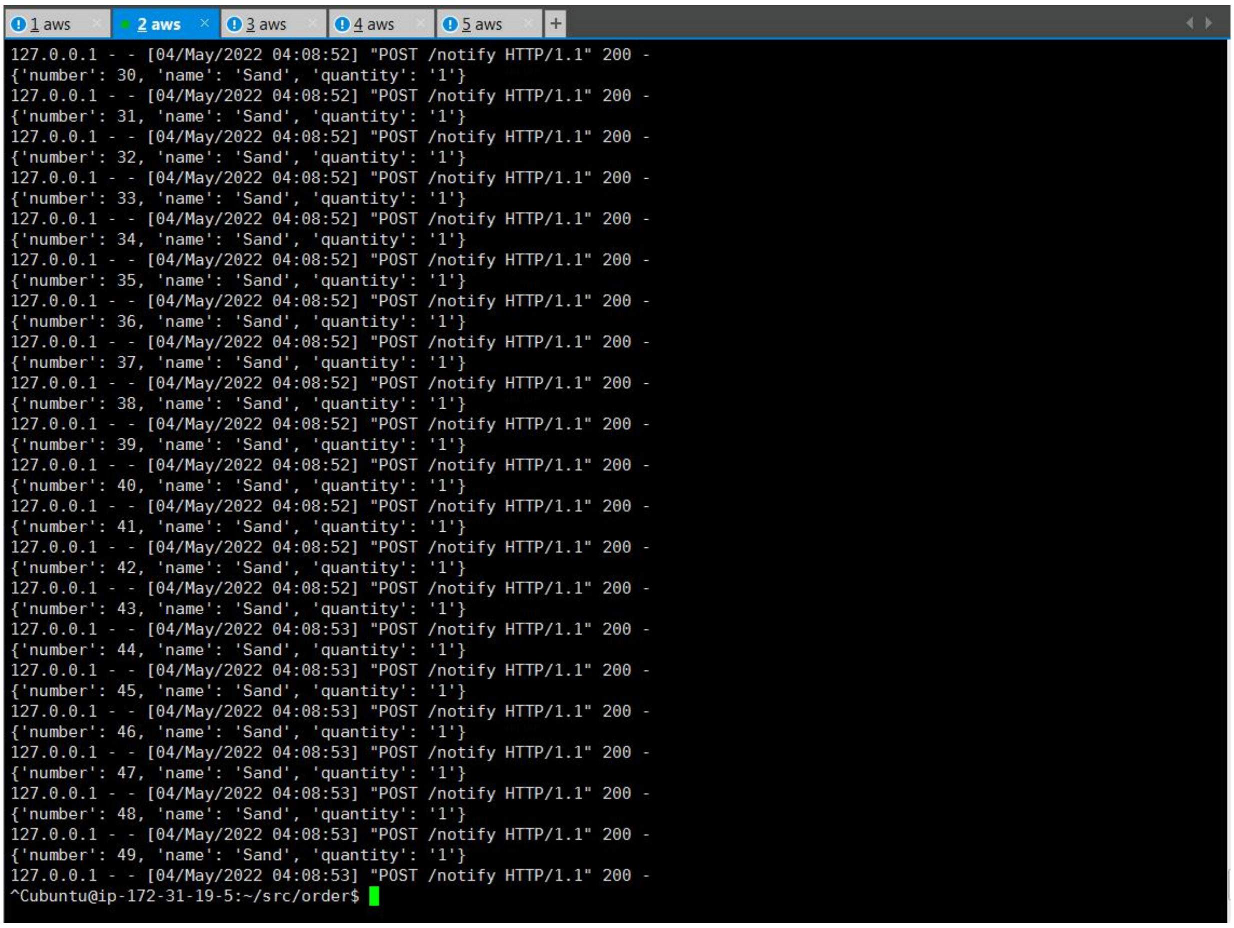
5. Fault Tolerance Test Output

Finally, we are simulating crash failures by killing a random order service replica while the clients is running, and then bring it back online after some time.

Client Terminal: we are sending **1000** buy requests using the code in load test.

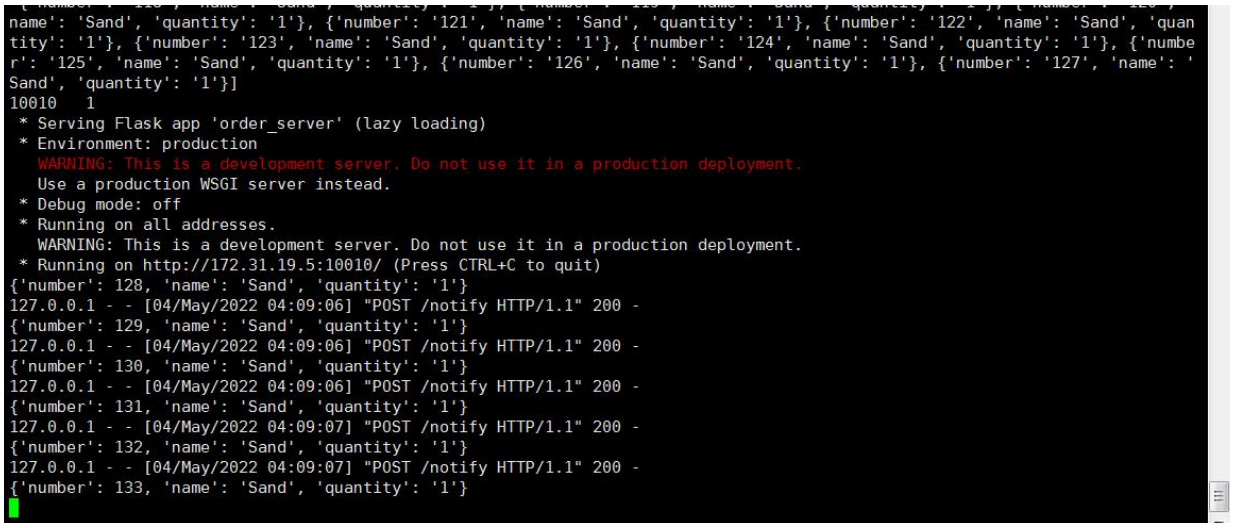
$ FRONT=<IP address> python3 -m unittest -v test_load.TestLoadPerformance.test_load_buy

Crash the follower with id = 1: terminate the node with **port = 10010** & **id=1**



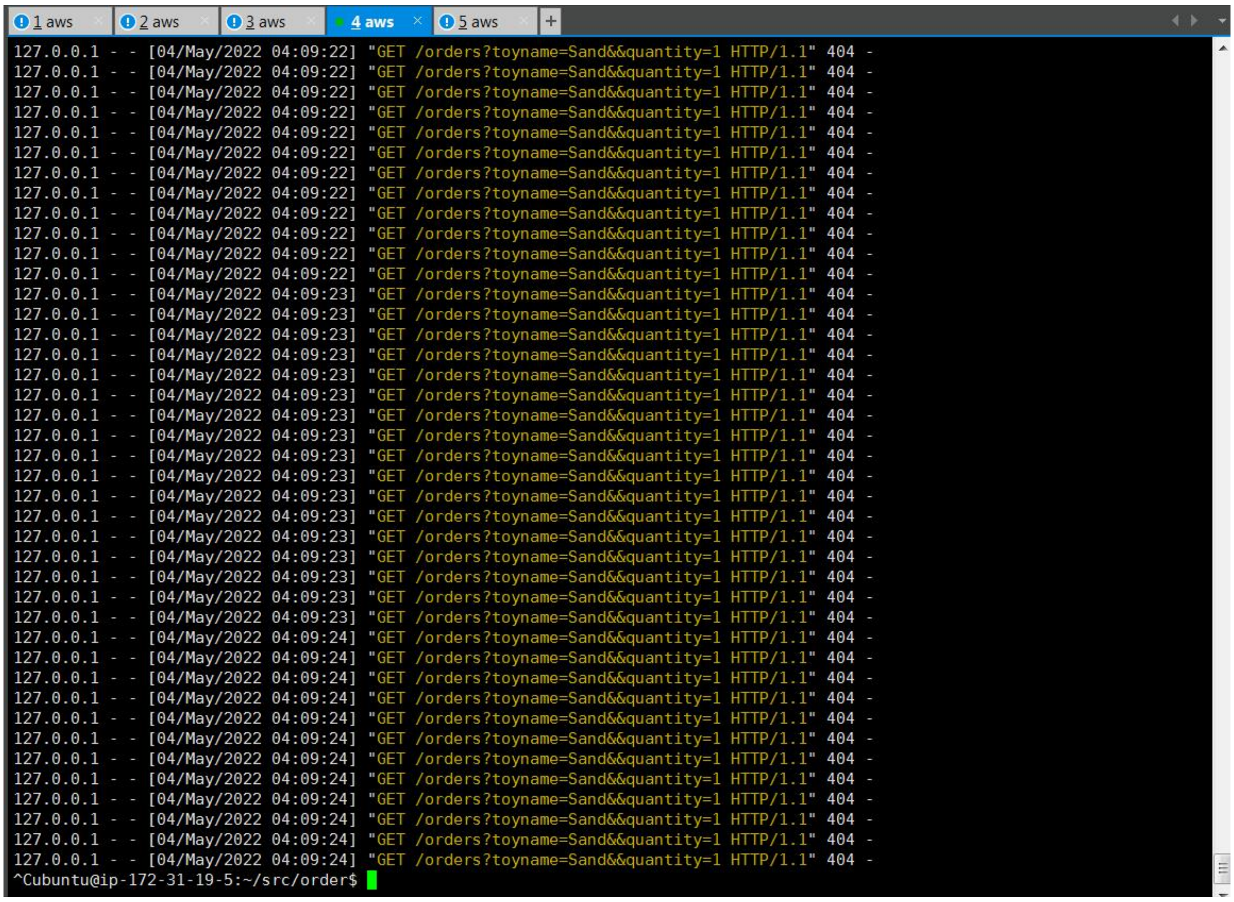
```
127.0.0.1 - - [04/May/2022 04:08:52] "POST /notify HTTP/1.1" 200 -
{'number': 30, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:52] "POST /notify HTTP/1.1" 200 -
{'number': 31, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:52] "POST /notify HTTP/1.1" 200 -
{'number': 32, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:52] "POST /notify HTTP/1.1" 200 -
{'number': 33, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:52] "POST /notify HTTP/1.1" 200 -
{'number': 34, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:52] "POST /notify HTTP/1.1" 200 -
{'number': 35, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:52] "POST /notify HTTP/1.1" 200 -
{'number': 36, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:52] "POST /notify HTTP/1.1" 200 -
{'number': 37, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:52] "POST /notify HTTP/1.1" 200 -
{'number': 38, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:52] "POST /notify HTTP/1.1" 200 -
{'number': 39, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:52] "POST /notify HTTP/1.1" 200 -
{'number': 40, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:52] "POST /notify HTTP/1.1" 200 -
{'number': 41, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:52] "POST /notify HTTP/1.1" 200 -
{'number': 42, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:52] "POST /notify HTTP/1.1" 200 -
{'number': 43, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:53] "POST /notify HTTP/1.1" 200 -
{'number': 44, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:53] "POST /notify HTTP/1.1" 200 -
{'number': 45, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:53] "POST /notify HTTP/1.1" 200 -
{'number': 46, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:53] "POST /notify HTTP/1.1" 200 -
{'number': 47, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:53] "POST /notify HTTP/1.1" 200 -
{'number': 48, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:53] "POST /notify HTTP/1.1" 200 -
{'number': 49, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:08:53] "POST /notify HTTP/1.1" 200 -
^Cubuntu@ip-172-31-19-5:~/src/order$
```

Restart the follower with id = 1: restart the node with **port = 10010** & **id=1**



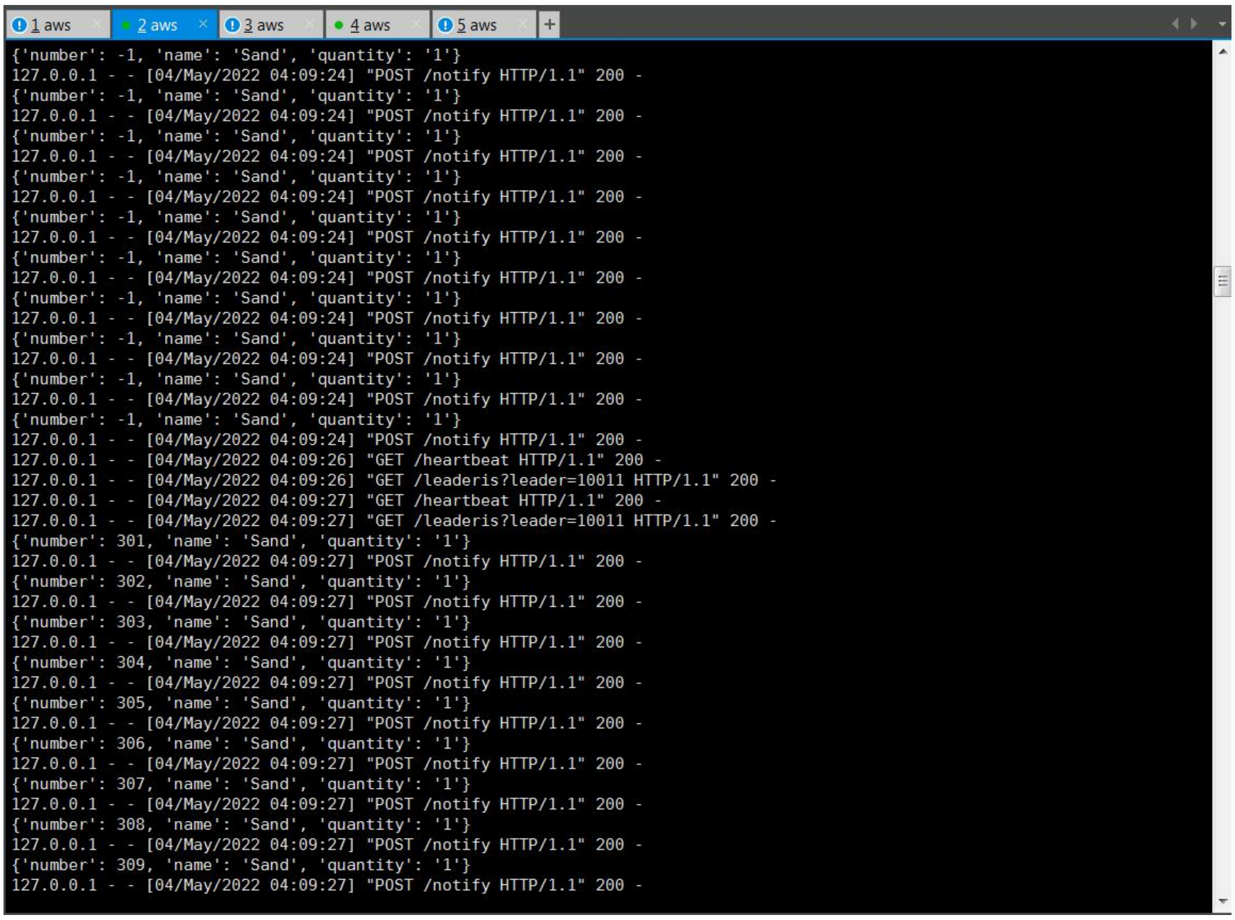
```
name': 'Sand', 'quantity': '1'}, {'number': '121', 'name': 'Sand', 'quantity': '1'}, {'number': '122', 'name': 'Sand', 'quan
tity': '1'}, {'number': '123', 'name': 'Sand', 'quantity': '1'}, {'number': '124', 'name': 'Sand', 'quantity': '1'}, {'numbe
r': '125', 'name': 'Sand', 'quantity': '1'}, {'number': '126', 'name': 'Sand', 'quantity': '1'}, {'number': '127', 'name': '
Sand', 'quantity': '1'}]
10010    1
 * Serving Flask app 'order_server' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.31.19.5:10010/ (Press CTRL+C to quit)
{'number': 128, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:06] "POST /notify HTTP/1.1" 200 -
{'number': 129, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:06] "POST /notify HTTP/1.1" 200 -
{'number': 130, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:06] "POST /notify HTTP/1.1" 200 -
{'number': 131, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:07] "POST /notify HTTP/1.1" 200 -
{'number': 132, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:07] "POST /notify HTTP/1.1" 200 -
{'number': 133, 'name': 'Sand', 'quantity': '1'}
```

Crash the leader with id = 3: terminate the node with **port = 10012** & **id=3**

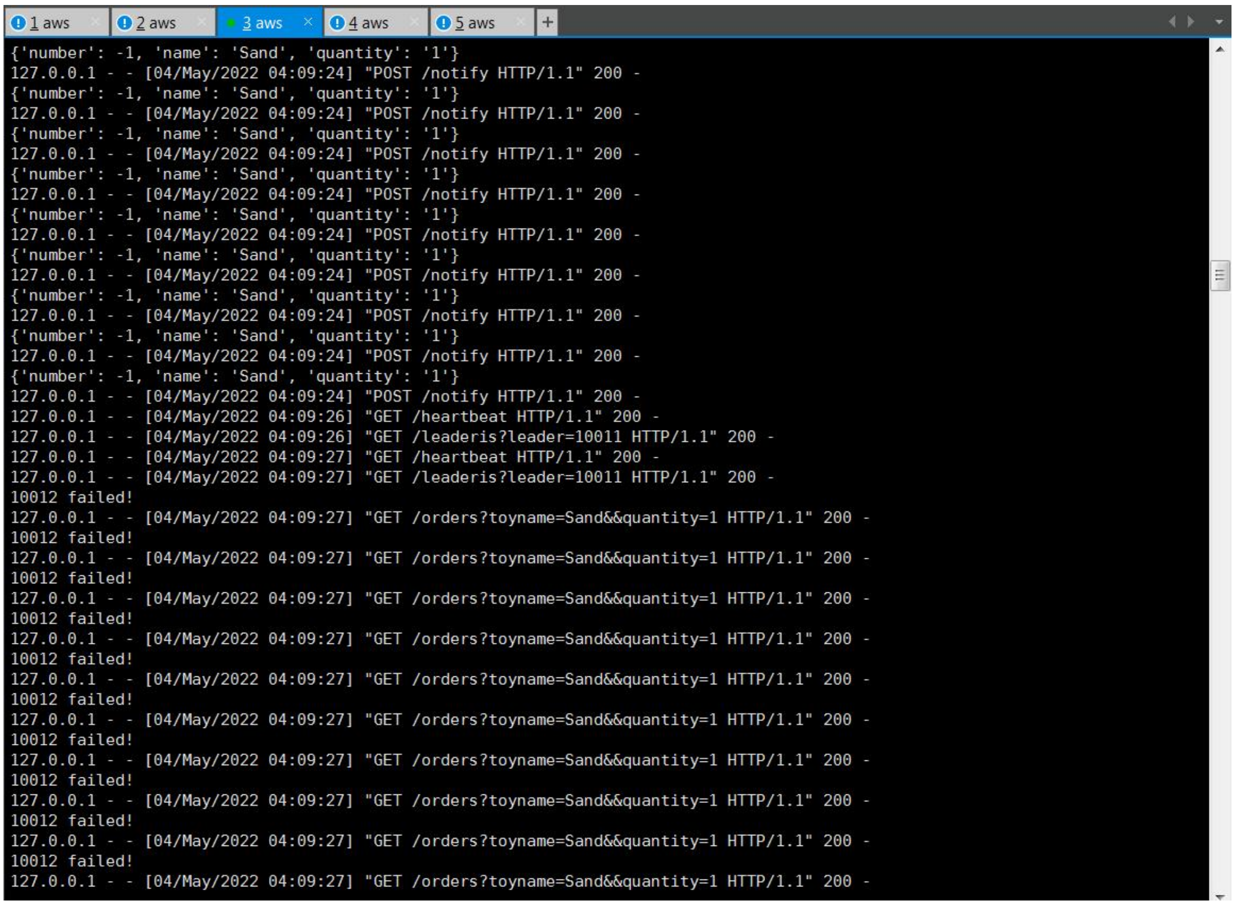


```
127.0.0.1 - - [04/May/2022 04:09:22] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:22] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:22] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:22] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:22] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:22] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:22] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:22] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:22] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:22] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:22] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:22] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:23] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:23] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:23] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:23] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:23] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:23] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:23] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:23] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:23] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:23] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:23] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:23] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:23] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:23] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:23] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:23] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:23] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:24] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:24] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:24] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:24] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:24] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:24] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:24] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:24] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:24] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:24] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:24] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
127.0.0.1 - - [04/May/2022 04:09:24] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 404 -
^Cubuntu@ip-172-31-19-5:~/src/order$
```

New leader notification: since leader is crashed, front end performed the **leader election**, and notify other nodes who is the new leader.

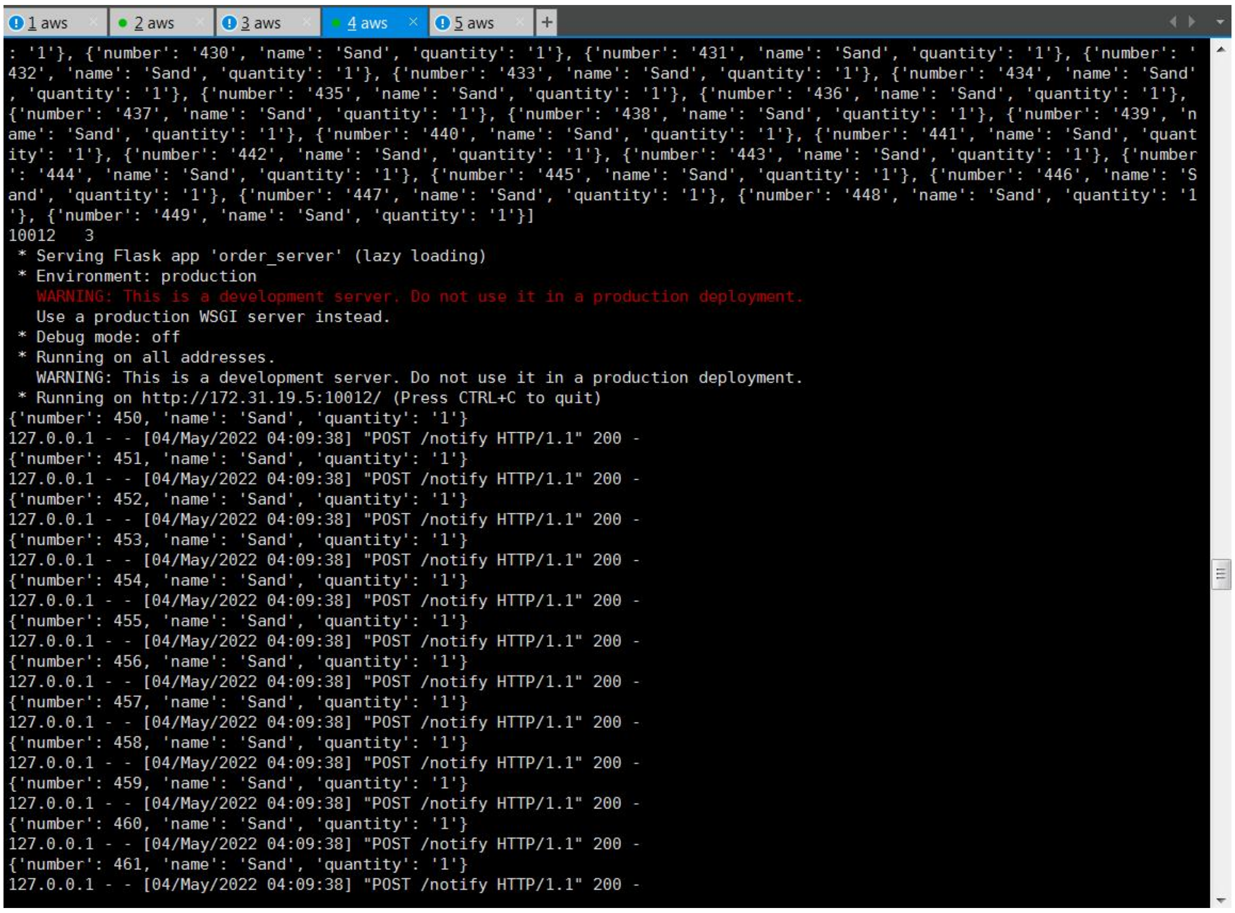


```
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
127.0.0.1 - - [04/May/2022 04:09:26] "GET /heartbeat HTTP/1.1" 200 -
127.0.0.1 - - [04/May/2022 04:09:26] "GET /leaderis?leader=10011 HTTP/1.1" 200 -
127.0.0.1 - - [04/May/2022 04:09:27] "GET /heartbeat HTTP/1.1" 200 -
127.0.0.1 - - [04/May/2022 04:09:27] "GET /leaderis?leader=10011 HTTP/1.1" 200 -
{'number': 301, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:27] "POST /notify HTTP/1.1" 200 -
{'number': 302, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:27] "POST /notify HTTP/1.1" 200 -
{'number': 303, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:27] "POST /notify HTTP/1.1" 200 -
{'number': 304, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:27] "POST /notify HTTP/1.1" 200 -
{'number': 305, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:27] "POST /notify HTTP/1.1" 200 -
{'number': 306, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:27] "POST /notify HTTP/1.1" 200 -
{'number': 307, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:27] "POST /notify HTTP/1.1" 200 -
{'number': 308, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:27] "POST /notify HTTP/1.1" 200 -
{'number': 309, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:27] "POST /notify HTTP/1.1" 200 -
```



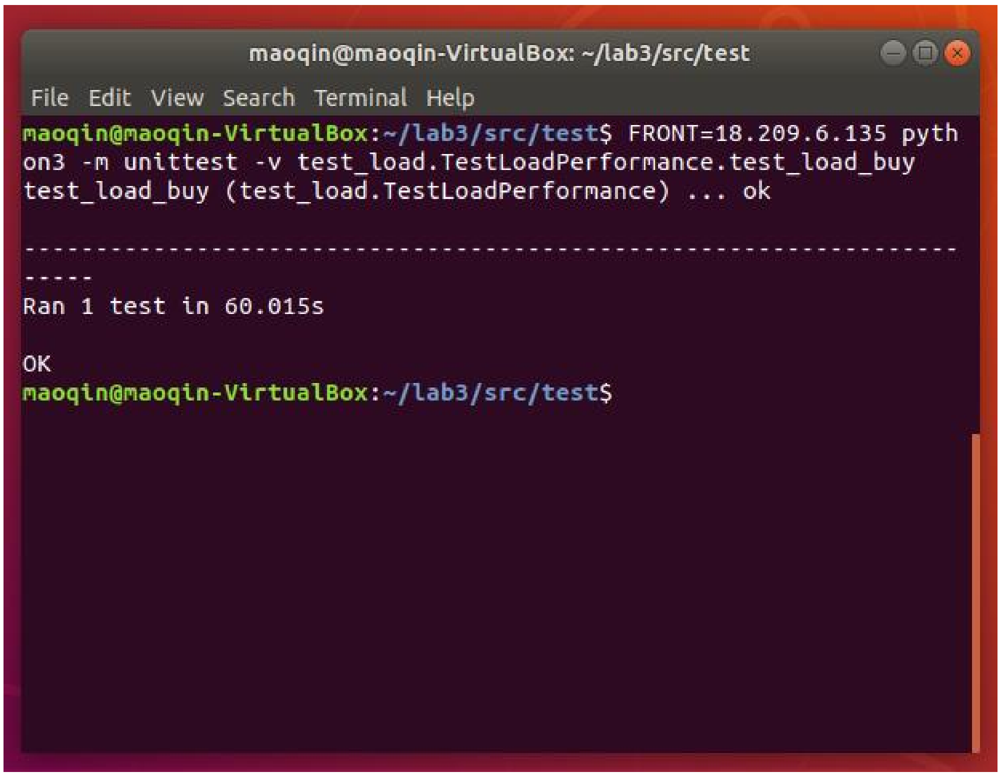
```
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
{'number': -1, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:24] "POST /notify HTTP/1.1" 200 -
127.0.0.1 - - [04/May/2022 04:09:26] "GET /heartbeat HTTP/1.1" 200 -
127.0.0.1 - - [04/May/2022 04:09:26] "GET /leaderis?leader=10011 HTTP/1.1" 200 -
127.0.0.1 - - [04/May/2022 04:09:27] "GET /heartbeat HTTP/1.1" 200 -
127.0.0.1 - - [04/May/2022 04:09:27] "GET /leaderis?leader=10011 HTTP/1.1" 200 -
10012 failed!
127.0.0.1 - - [04/May/2022 04:09:27] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 200 -
10012 failed!
127.0.0.1 - - [04/May/2022 04:09:27] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 200 -
10012 failed!
127.0.0.1 - - [04/May/2022 04:09:27] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 200 -
10012 failed!
127.0.0.1 - - [04/May/2022 04:09:27] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 200 -
10012 failed!
127.0.0.1 - - [04/May/2022 04:09:27] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 200 -
10012 failed!
127.0.0.1 - - [04/May/2022 04:09:27] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 200 -
10012 failed!
127.0.0.1 - - [04/May/2022 04:09:27] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 200 -
10012 failed!
127.0.0.1 - - [04/May/2022 04:09:27] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 200 -
10012 failed!
127.0.0.1 - - [04/May/2022 04:09:27] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 200 -
10012 failed!
127.0.0.1 - - [04/May/2022 04:09:27] "GET /orders?toyname=Sand&&quantity=1 HTTP/1.1" 200 -
```

Restart the follower with id = 3: restart the node with **port = 10012** & **id=3**



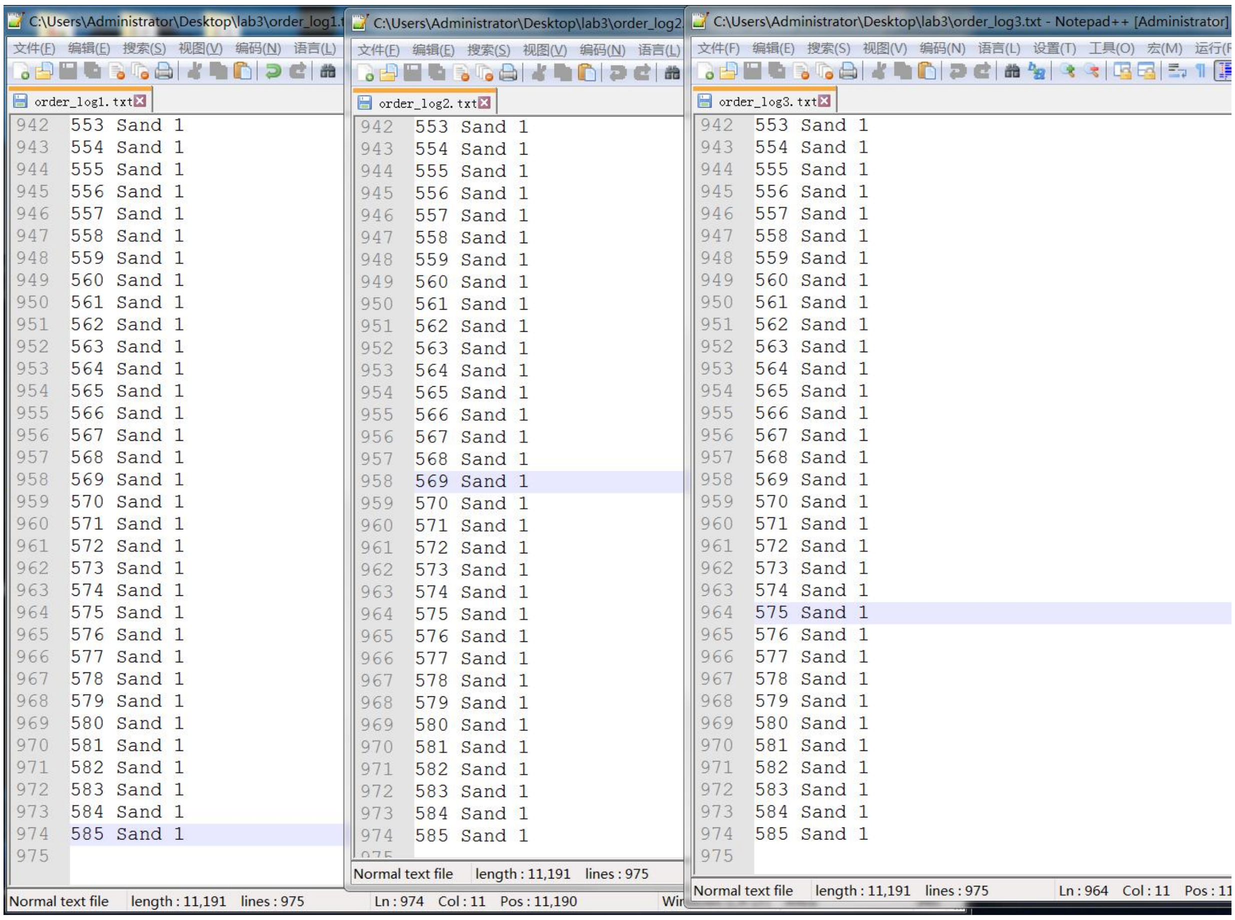
```
: '1'}, {'number': '430', 'name': 'Sand', 'quantity': '1'}, {'number': '431', 'name': 'Sand', 'quantity': '1'}, {'number': '
432', 'name': 'Sand', 'quantity': '1'}, {'number': '433', 'name': 'Sand', 'quantity': '1'}, {'number': '434', 'name': 'Sand'
, 'quantity': '1'}, {'number': '435', 'name': 'Sand', 'quantity': '1'}, {'number': '436', 'name': 'Sand', 'quantity': '1'},
{'number': '437', 'name': 'Sand', 'quantity': '1'}, {'number': '438', 'name': 'Sand', 'quantity': '1'}, {'number': '439', 'n
ame': 'Sand', 'quantity': '1'}, {'number': '440', 'name': 'Sand', 'quantity': '1'}, {'number': '441', 'name': 'Sand', 'quant
ity': '1'}, {'number': '442', 'name': 'Sand', 'quantity': '1'}, {'number': '443', 'name': 'Sand', 'quantity': '1'}, {'number
': '444', 'name': 'Sand', 'quantity': '1'}, {'number': '445', 'name': 'Sand', 'quantity': '1'}, {'number': '446', 'name': 'S
and', 'quantity': '1'}, {'number': '447', 'name': 'Sand', 'quantity': '1'}, {'number': '448', 'name': 'Sand', 'quantity': '1
'}, {'number': '449', 'name': 'Sand', 'quantity': '1'}]
10012   3
 * Serving Flask app 'order_server' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.31.19.5:10012/ (Press CTRL+C to quit)
{'number': 450, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:38] "POST /notify HTTP/1.1" 200 -
{'number': 451, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:38] "POST /notify HTTP/1.1" 200 -
{'number': 452, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:38] "POST /notify HTTP/1.1" 200 -
{'number': 453, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:38] "POST /notify HTTP/1.1" 200 -
{'number': 454, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:38] "POST /notify HTTP/1.1" 200 -
{'number': 455, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:38] "POST /notify HTTP/1.1" 200 -
{'number': 456, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:38] "POST /notify HTTP/1.1" 200 -
{'number': 457, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:38] "POST /notify HTTP/1.1" 200 -
{'number': 458, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:38] "POST /notify HTTP/1.1" 200 -
{'number': 459, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:38] "POST /notify HTTP/1.1" 200 -
{'number': 460, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:38] "POST /notify HTTP/1.1" 200 -
{'number': 461, 'name': 'Sand', 'quantity': '1'}
127.0.0.1 - - [04/May/2022 04:09:38] "POST /notify HTTP/1.1" 200 -
```

Total latency seen by clients:


maoqin@maoqin-VirtualBox: ~/lab3/src/test
File Edit View Search Terminal Help

```
maoqin@maoqin-VirtualBox:~/lab3/src/test$ FRONT=18.209.6.135 pyth
on3 -m unittest -v test_load.TestLoadPerformance.test_load_buy
test_load_buy (test_load.TestLoadPerformance) ... ok

----------------------------------------------------------------------
-----
Ran 1 test in 60.015s

OK
maoqin@maoqin-VirtualBox:~/lab3/src/test$
```

Order log at each order server:


文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O) 宏(M) 运行(R)

C:\Users\Administrator\Desktop\lab3\order_log1.t

order_log1.txt

```
553 Sand 1
554 Sand 1
555 Sand 1
556 Sand 1
557 Sand 1
558 Sand 1
559 Sand 1
560 Sand 1
561 Sand 1
562 Sand 1
563 Sand 1
564 Sand 1
565 Sand 1
566 Sand 1
567 Sand 1
568 Sand 1
569 Sand 1
570 Sand 1
571 Sand 1
572 Sand 1
573 Sand 1
574 Sand 1
575 Sand 1
576 Sand 1
577 Sand 1
578 Sand 1
579 Sand 1
580 Sand 1
581 Sand 1
582 Sand 1
583 Sand 1
584 Sand 1
585 Sand 1
```

Normal text file   length : 11,191   lines : 975   Ln : 974   Col : 11   Pos : 11,190   Wi

C:\Users\Administrator\Desktop\lab3\order_log2

order_log2.txt

```
553 Sand 1
554 Sand 1
555 Sand 1
556 Sand 1
557 Sand 1
558 Sand 1
559 Sand 1
560 Sand 1
561 Sand 1
562 Sand 1
563 Sand 1
564 Sand 1
565 Sand 1
566 Sand 1
567 Sand 1
568 Sand 1
569 Sand 1
570 Sand 1
571 Sand 1
572 Sand 1
573 Sand 1
574 Sand 1
575 Sand 1
576 Sand 1
577 Sand 1
578 Sand 1
579 Sand 1
580 Sand 1
581 Sand 1
582 Sand 1
583 Sand 1
584 Sand 1
585 Sand 1
```

Normal text file   length : 11,191   lines : 975

C:\Users\Administrator\Desktop\lab3\order_log3.txt - Notepad++ [Administrator]

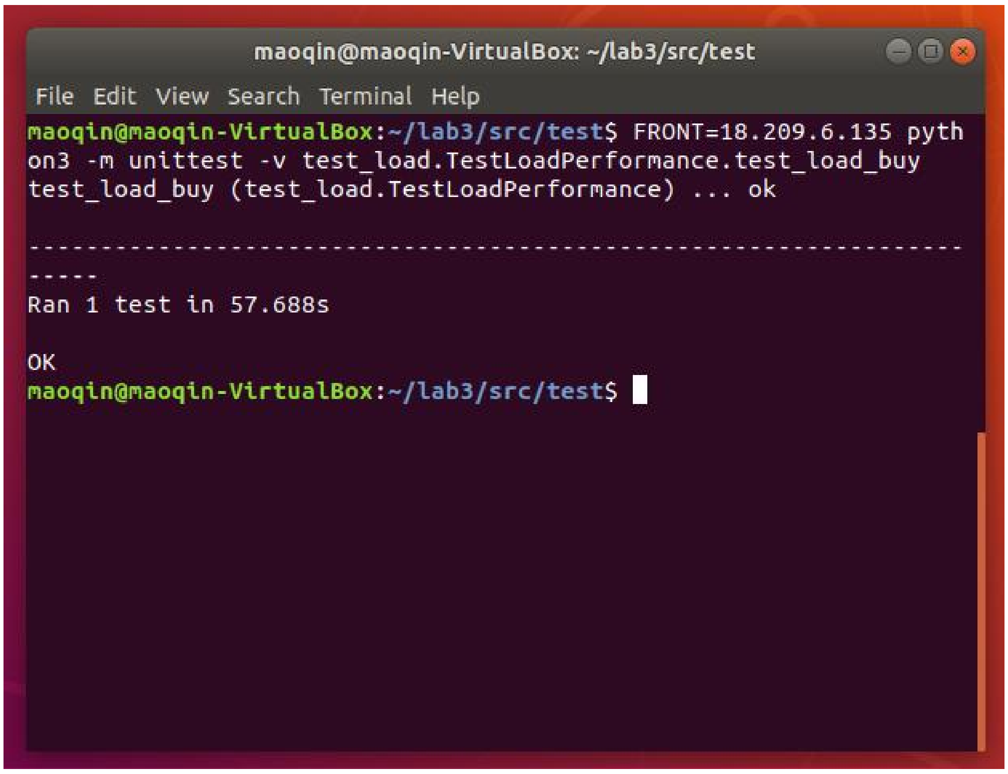
order_log3.txt

```
553 Sand 1
554 Sand 1
555 Sand 1
556 Sand 1
557 Sand 1
558 Sand 1
559 Sand 1
560 Sand 1
561 Sand 1
562 Sand 1
563 Sand 1
564 Sand 1
565 Sand 1
566 Sand 1
567 Sand 1
568 Sand 1
569 Sand 1
570 Sand 1
571 Sand 1
572 Sand 1
573 Sand 1
574 Sand 1
575 Sand 1
576 Sand 1
577 Sand 1
578 Sand 1
579 Sand 1
580 Sand 1
581 Sand 1
582 Sand 1
583 Sand 1
584 Sand 1
585 Sand 1
```

Normal text file   length : 11,191   lines : 975   Ln : 964   Col : 11   Pos : 11

In order to evaluate in what degree the clients can notice the failure, we do the same experiment without artificial crashes.

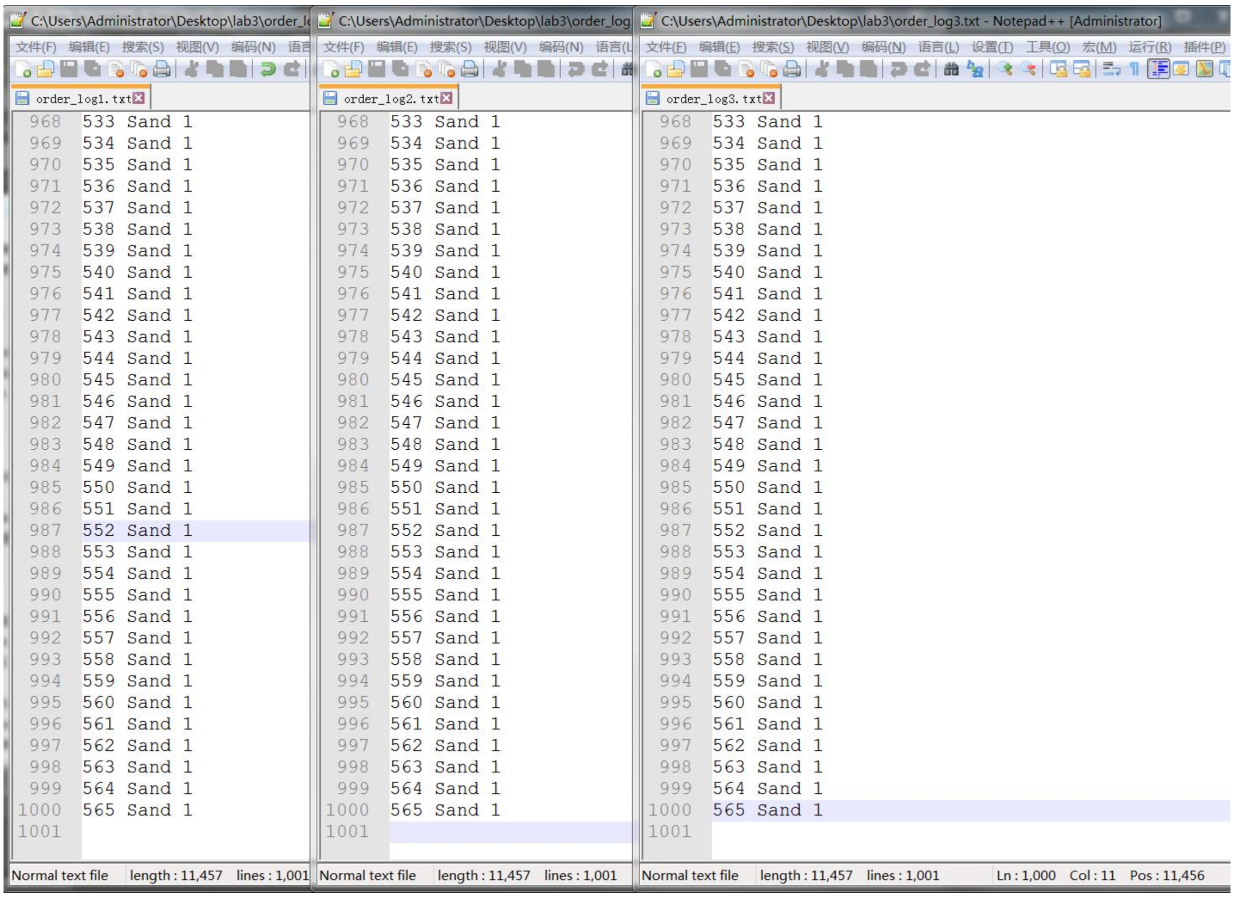
Total latency seen by clients without artificial crashes:



maoqin@maoqin-VirtualBox: ~/lab3/src/test
File Edit View Search Terminal Help

```
maoqin@maoqin-VirtualBox:~/lab3/src/test$ FRONT=18.209.6.135 pyth
on3 -m unittest -v test_load.TestLoadPerformance.test_load_buy
test_load_buy (test_load.TestLoadPerformance) ... ok

----------------------------------------------------------------------
-----
Ran 1 test in 57.688s

OK
maoqin@maoqin-VirtualBox:~/lab3/src/test$ 
```

Order log at each order server without artificial crashes:



C:\Users\Administrator\Desktop\lab3\order_l
C:\Users\Administrator\Desktop\lab3\order_log
C:\Users\Administrator\Desktop\lab3\order_log3.txt - Notepad++ [Administrator]
文件(F) 编辑(E) 搜索(S) 视图(V) 编码(N) 语言(L) 设置(T) 工具(O) 宏(M) 运行(R) 插件(P)
order_log1.txt
order_log2.txt
order_log3.txt

```
533 Sand 1
534 Sand 1
535 Sand 1
536 Sand 1
537 Sand 1
538 Sand 1
539 Sand 1
540 Sand 1
541 Sand 1
542 Sand 1
543 Sand 1
544 Sand 1
545 Sand 1
546 Sand 1
547 Sand 1
548 Sand 1
549 Sand 1
550 Sand 1
551 Sand 1
552 Sand 1
553 Sand 1
554 Sand 1
555 Sand 1
556 Sand 1
557 Sand 1
558 Sand 1
559 Sand 1
560 Sand 1
561 Sand 1
562 Sand 1
563 Sand 1
564 Sand 1
565 Sand 1

```

```
533 Sand 1
534 Sand 1
535 Sand 1
536 Sand 1
537 Sand 1
538 Sand 1
539 Sand 1
540 Sand 1
541 Sand 1
542 Sand 1
543 Sand 1
544 Sand 1
545 Sand 1
546 Sand 1
547 Sand 1
548 Sand 1
549 Sand 1
550 Sand 1
551 Sand 1
552 Sand 1
553 Sand 1
554 Sand 1
555 Sand 1
556 Sand 1
557 Sand 1
558 Sand 1
559 Sand 1
560 Sand 1
561 Sand 1
562 Sand 1
563 Sand 1
564 Sand 1
565 Sand 1

```

```
533 Sand 1
534 Sand 1
535 Sand 1
536 Sand 1
537 Sand 1
538 Sand 1
539 Sand 1
540 Sand 1
541 Sand 1
542 Sand 1
543 Sand 1
544 Sand 1
545 Sand 1
546 Sand 1
547 Sand 1
548 Sand 1
549 Sand 1
550 Sand 1
551 Sand 1
552 Sand 1
553 Sand 1
554 Sand 1
555 Sand 1
556 Sand 1
557 Sand 1
558 Sand 1
559 Sand 1
560 Sand 1
561 Sand 1
562 Sand 1
563 Sand 1
564 Sand 1
565 Sand 1

```

Normal text file length : 11,457 lines : 1,001 | Normal text file length : 11,457 lines : 1,001 | Normal text file length : 11,457 lines : 1,001 Ln : 1,000 Col : 11 Pos : 11,456