

## Outputs:

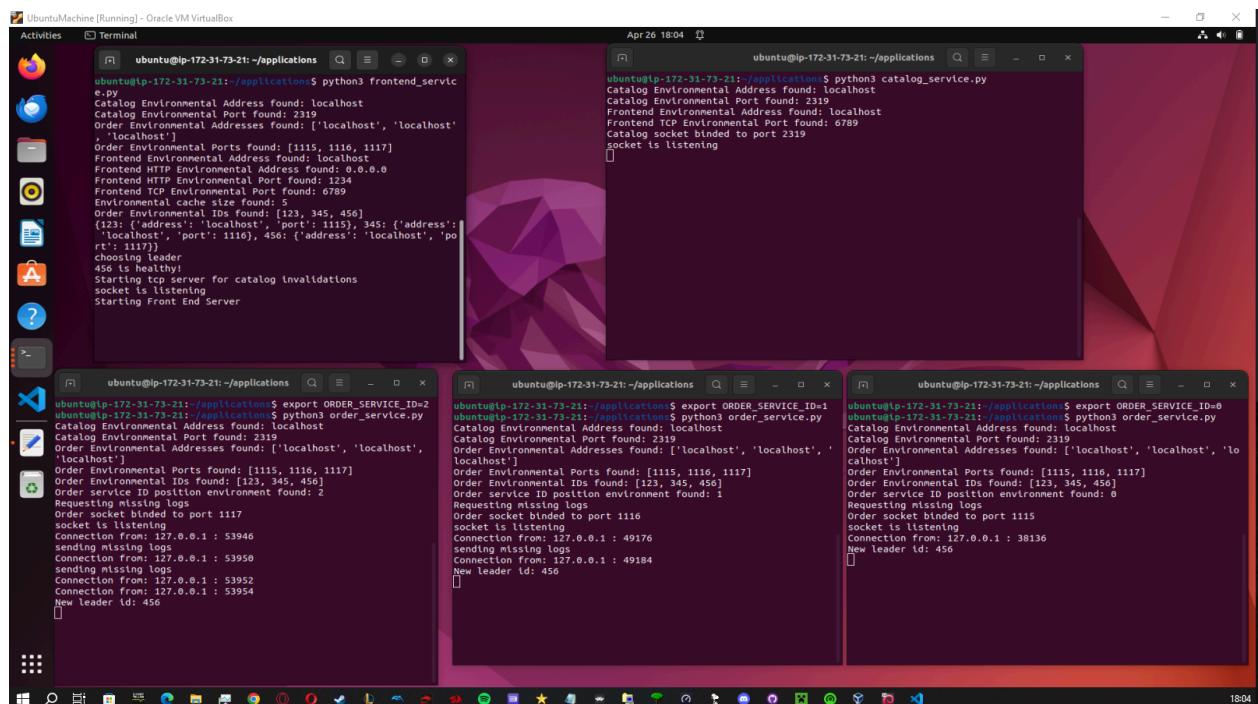
These outputs are generated from a Ubuntu virtual machine which has used an SSH connection to connect to the AWS machine. Each terminal is using the SSH connection to access and start the Frontend service, the Catalog service, and the Order service replicas.

The Ubuntu machine screenshots are organized as:

Top Left: Frontend service. Top Right: Catalog service. Bottom Left: Order service replica ID 456. Bottom Middle: Order service replica ID 345. Bottom Right: Order service replica ID 123.

The client is run on a Kali Linux virtual machine and the client is connecting to the AWS server.

After connecting by SSH to the AWS server, Initializing the services on the AWS server:



The image shows four terminal windows running on an Ubuntu desktop environment. The desktop interface includes a dock with icons for various applications like a browser, file manager, and terminal. The terminals are arranged in a 2x2 grid. The top-left terminal shows the output of starting the Frontend service. The top-right terminal shows the output of starting the Catalog service. The bottom-left terminal shows the output of starting Order service replica ID 456. The bottom-middle terminal shows the output of starting Order service replica ID 345. All terminals show environmental variable assignments and socket binding logs.

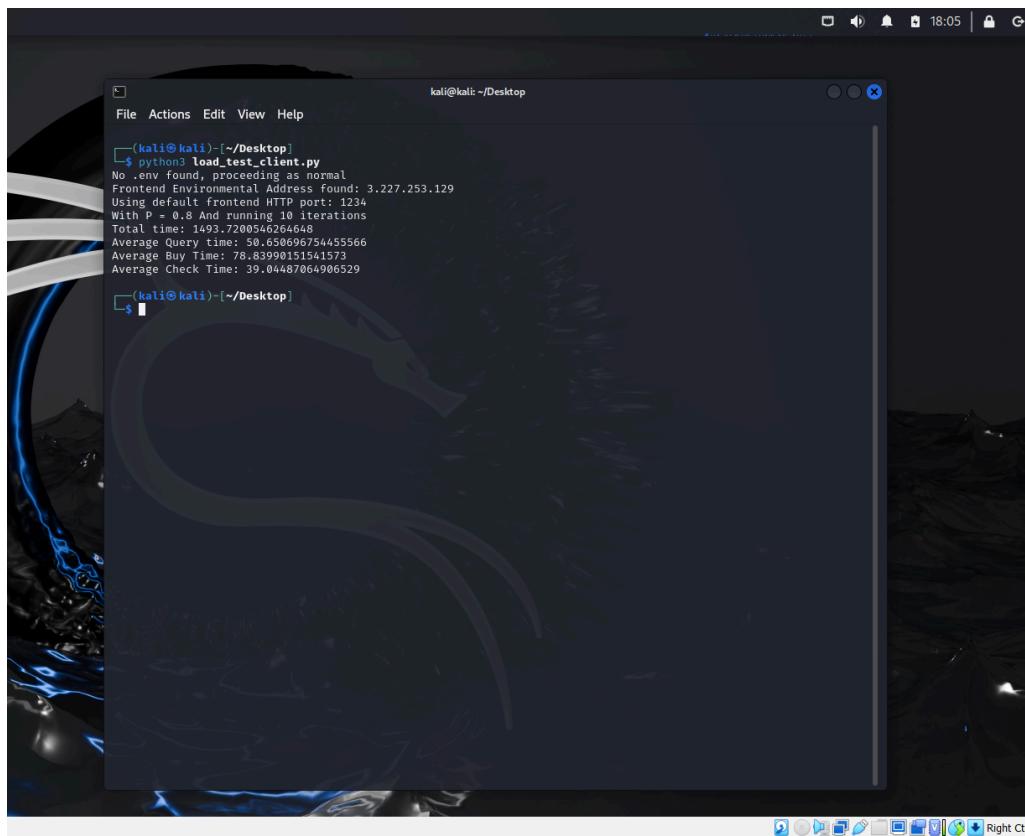
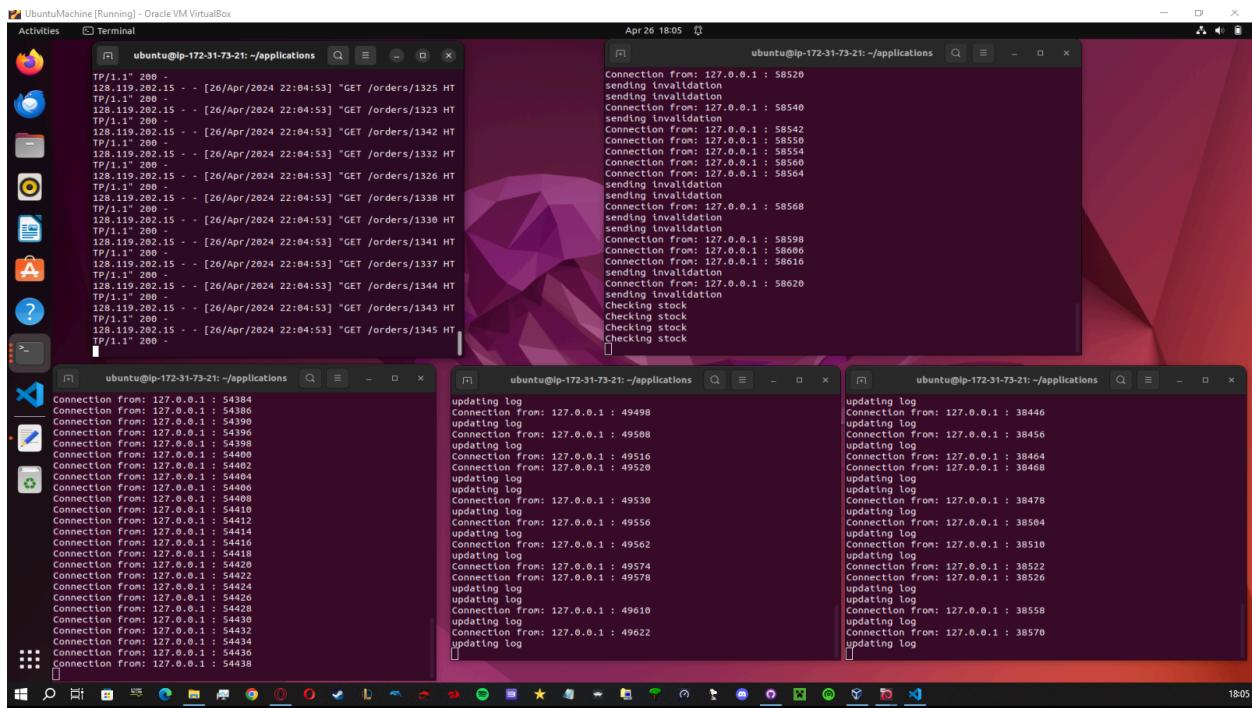
```
ubuntu@ip-172-31-73-21:~/applications$ python3 frontend_service.py
...
Catalog Environmental Address found: localhost
Catalog Environmental Port found: 2319
Order Environmental Addresses found: ['localhost', 'localhost']
...
Order Environmental Ports found: [1115, 1116, 1117]
Frontend Environmental Address found: localhost
Frontend HTTP Environmental Address found: 0.0.0.0
Frontend HTTP Environmental Port found: 1234
Frontend TCP Environmental Address found: 6789
Environmental Cache size found: 5
Order Environmental IDs found: [123, 345, 456]
[123: {'address': 'localhost', 'port': 1115}, 345: {'address': 'localhost', 'port': 1116}, 456: {'address': 'localhost', 'port': 1117}]
choosing Leader
456 is healthy!
Starting tcp server for catalog invalidations
socket is listening
Starting Front End Server

ubuntu@ip-172-31-73-21:~/applications$ python3 catalog_service.py
...
Catalog Environmental Address found: localhost
Catalog Environmental Port found: 2319
Frontend Environmental Address found: localhost
Frontend TCP Environmental Port found: 6789
Catalog socket binded to port 2319
socket is listening

ubuntu@ip-172-31-73-21:~/applications$ export ORDER_SERVICE_ID=1
ubuntu@ip-172-31-73-21:~/applications$ python3 order_service.py
...
Catalog Environmental Address found: localhost
Catalog Environmental Port found: 2319
Order Environmental Addresses found: ['localhost', 'localhost', 'localhost']
...
Order Environmental Ports found: [1115, 1116, 1117]
Order Environmental IDs found: [123, 345, 456]
Order service ID position environment found: 2
Requesting missing logs
Order socket binded to port 1117
socket is listening
Connection from: 127.0.0.1 : 53946
sending missing logs
Connection from: 127.0.0.1 : 53950
sending missing logs
Connection from: 127.0.0.1 : 53952
Connection from: 127.0.0.1 : 53954
New leader id: 456

ubuntu@ip-172-31-73-21:~/applications$ export ORDER_SERVICE_ID=0
ubuntu@ip-172-31-73-21:~/applications$ python3 order_service.py
...
Catalog Environmental Address found: localhost
Catalog Environmental Port found: 2319
Order Environmental Addresses found: ['localhost', 'localhost', 'localhost']
...
Order Environmental Ports found: [1115, 1116, 1117]
Order Environmental IDs found: [123, 345, 456]
Order service ID position environment found: 1
Requesting missing logs
Order socket binded to port 1116
socket is listening
Connection from: 127.0.0.1 : 49176
sending missing logs
Connection from: 127.0.0.1 : 49184
New leader id: 456
```

Under normal conditions with the services running on AWS performing the load test client:



Using the Human version of the client to see transparency. Placing a normal order. Notice the catalog service sending the cache invalidation to the frontend for that order:

The screenshot shows four terminal windows on an Ubuntu desktop environment. The windows are arranged in a grid-like fashion. The top row contains two windows, and the bottom row contains two windows. Each window displays log output from a different service.

- Top Left Window:** Shows logs for the `catalog_service.py` script. It includes environmental variable definitions like `CATALOG_HOST=127.0.0.1`, port assignments (2319 for Catalog Env, 6789 for Frontend Env), and a message about sending a cache invalidation to the frontend.
- Top Right Window:** Shows logs for the `order_service.py` script. It includes environmental variable definitions like `ORDER_HOST=127.0.0.1`, port assignments (2319 for Catalog Env, 47330 for Order Env), and a message about updating the log after becoming the new leader.
- Bottom Left Window:** Shows logs for the `frontend_service.py` script. It includes environmental variable definitions like `FRONTEND_HOST=127.0.0.1`, port assignments (2319 for Catalog Env, 1115 for Frontend Env), and a message about sending missing logs to the catalog service.
- Bottom Right Window:** Shows logs for the `order_client.py` script. It includes environmental variable definitions like `ORDER_SERVICE_ID=2`, port assignments (2319 for Catalog Env, 1115 for Order Env), and a message about attempting to buy an item with ID 1903.

The screenshot shows a single terminal window on a Kali Linux desktop environment. The window displays the interaction with the `order_client.py` script. The user enters commands to query, buy, and check the status of an order. The script handles these requests and provides feedback, such as identifying the current leader and sending logs to the catalog service.

```
(kali㉿kali)-[~/Desktop]
$ python3 client.py
No .env found, proceeding as normal
Frontend Environmental Address found: 3.227.253.129
Using default Frontend HTTP port: 1234
Q: Query
B: Buy
O: Check Order
b
What product do you want?
whale
How many?
2
Attempting to buy
{'data': {'order_number': 1903}}
Q: Query
B: Buy
O: Check Order
```

Killing the lead order service (bottom left) and attempting to place another order.

You can see the new leader election process happening in the services, and the client never sees

the downed replica:

The image shows four terminal windows on an Ubuntu desktop. The top-left window shows a client attempting to place an order to the lead order service (Service ID 456). The client receives a response indicating Service ID 456 is down. The top-right window shows the catalog service performing leader election, identifying Service ID 456 as unhealthy and Service ID 345 as healthy. The bottom-left window shows the order service performing leader election, identifying Service ID 456 as down and Service ID 345 as up. The bottom-right window shows the catalog service continuing its leader election process after the order service has become the new leader.

```
ubuntu@ip-172-31-73-21:~/applications$ python3 catalog_service.py
Catalog Environmental Address found: localhost
Catalog Environmental Port Found: 2319
Frontend TCP Environmental Address found: localhost
Frontend TCP Environmental Port Found: 6789
Catalog socket binded to port 2319
Catalog is listening
Checking stock
Connection from: 127.0.0.1 : 39170
sending invalidation
Checking stock
Connection from: 127.0.0.1 : 39192
sending invalidation
Checking stock
Checking stock
Connection from: 127.0.0.1 : 50778
Connection from: 127.0.0.1 - - [26/Apr/2024 22:21:55] "POST /orders HTTP/1.1" 200 -
Service ID 456 is down.
Choosing leader
Service ID 456 is down.
345 is healthy!
Tried to alert Service ID of leader but 456 is down.
Connection from: 127.0.0.1 : 50800
[...]
ubuntu@ip-172-31-73-21:~/applications$ export ORDER_SERVICE_ID=456
ubuntu@ip-172-31-73-21:~/applications$ python3 order_service.py
Catalog Environmental Address found: localhost
Catalog Environmental Port Found: 2319
Order Environmental Addresses found: ['localhost', 'localhost', 'localhost']
Order Environmental Ports found: [1115, 1116, 1117]
Order Environmental IDs found: [123, 345, 456]
Order service ID position environment found: 1
Requesting missing logs
Order socket binded to port 1116
Socket is listening
Connection from: 127.0.0.1 : 58370
sending missing logs
Connection from: 127.0.0.1 : 58378
New leader id: 456
Updating log
Connection from: 127.0.0.1 : 58390
New leader id: 345
Updating log
Connection from: 127.0.0.1 : 58396
Connection from: 127.0.0.1 : 58400
Connection from: 127.0.0.1 : 58404
New leader id: 345
Updating log
[...]
ubuntu@ip-172-31-73-21:~/applications$ export ORDER_SERVICE_ID=0
ubuntu@ip-172-31-73-21:~/applications$ python3 catalog_service.py
Catalog Environmental Address found: localhost
Catalog Environmental Port Found: 2319
Order Environmental Addresses found: ['localhost', 'localhost', 'localhost']
Order Environmental Ports found: [1115, 1116, 1117]
Order Environmental IDs found: [123, 345, 456]
Order service ID position environment found: 0
Requesting missing logs
Order socket binded to port 1115
Socket is listening
Connection from: 127.0.0.1 : 47330
New leader id: 456
Connection from: 127.0.0.1 : 47338
Updating log
Connection from: 127.0.0.1 : 47352
New leader id: 345
Connection from: 127.0.0.1 : 47360
Updating log
[...]
```

The image shows a terminal window on a Kali Linux desktop. A client application is running, interacting with the system. It asks for a product, specifies an amount of 3, and attempts to buy it. The system responds with a success message, indicating the transaction was processed.

```
kali㉿kali:~/Desktop$ ./client.py
No .env found, proceeding as normal
Frontend Environmental Address found: 3.227.253.129
Using default frontend HTTP port: 1234
Q: Query
B: Buy
O: Check Order
b
What product do you want?
whole
How many?
Attempting to buy
{'data': {'order_number': 1903}}
Q: Query
B: Buy
O: Check Order
b
What product do you want?
tux
How many?
3
Attempting to buy
{'data': {'order_number': 1904}}
Q: Query
B: Buy
O: Check Order
[...]
```

With the original leader that processed order #1903 still down, check the order and see if the new leader kept up with the orders. Looking at the client you can see the response from the check order returns the same information as the order placed, despite the leader that originally processed the order being down:

```

UbuntuMachine [Running] - Oracle VM VirtualBox
Activities Terminal ubuntu@ip-172-31-73-21: ~applications Apr 26 16:22
Frontend HTTP Environmental Port found: 1234
Frontend TCP Environmental Port found: 6789
Environmental cache size found: 5
Order Environmental IDs found: [123, 345, 456]
[123: {'address': 'localhost', 'port': 1115}, 345: {'address': 'localhost', 'port': 1116}, 456: {'address': 'localhost', 'port': 1117}]
choosing leader
456 ls healthy
Starting Catalog server for catalog invalidations
socket is listening
Starting Front End Server
128.119.202.15 - - [26/Apr/2024 22:21:55] "POST /orders HTTP/1.1" 200
Connection from: 127.0.0.1 : 50778
128.119.202.15 - - [26/Apr/2024 22:22:18] "POST /orders HTTP/1.1" 200
Service ID 456 is down.
choosing leader
Service ID 456 is down.
345 is healthy!
Tried to alert Service ID of leader but 456 is down.
Connection from: 127.0.0.1 : 50800
128.119.202.15 - - [26/Apr/2024 22:22:46] "GET /orders/1903 HTTP/1.1" 200

ubuntu@ip-172-31-73-21: ~applications python3 catalog_service.py
Catalog Environmental Address Found: localhost
Catalog Environmental Port Found: 2319
Frontend Environmental Address Found: localhost
Frontend TCP Environmental Port Found: 6789
Catalog socket binded to port 2319
socket is listening
Checking stock
Checking stock
Connection from: 127.0.0.1 : 39170
sending Invalidations
Checking stock
Checking stock
Connection from: 127.0.0.1 : 39192
sending Invalidations
Checking stock
Checking stock
Checking stock
Checking stock

ubuntu@ip-172-31-73-21: ~applications python3 order_service.py
Order Environmental Ports found: [1115, 1116, 1117]
Order Environmental IDs found: [123, 345, 456]
Order service ID position environment found: 1
Reporting pending logs
Order socket binded to port 1115
socket is listening
Connection from: 127.0.0.1 : 58370
sending missing logs
Connection from: 127.0.0.1 : 58378
New leader id: 456
Connection from: 127.0.0.1 : 58390
updating log
Connection from: 127.0.0.1 : 58396
Connection from: 127.0.0.1 : 58400
Connection from: 127.0.0.1 : 58404
New leader id: 345
Connection from: 127.0.0.1 : 58414

ubuntu@ip-172-31-73-21: ~applications$ export ORDER_SERVICE_ID=345
ubuntu@ip-172-31-73-21: ~applications$ python3 order_service.py
Catalog Environmental Address Found: localhost
Catalog Environmental Port Found: 2319
Order Environmental Addresses found: ['localhost', 'localhost', 'localhost']
Order Environmental Ports found: [1115, 1116, 1117]
Order Environmental IDs found: [123, 345, 456]
Order service ID position environment found: 0
Reporting pending logs
Order socket binded to port 1115
socket is listening
Connection from: 127.0.0.1 : 47330
New leader id: 456
Connection from: 127.0.0.1 : 47338
updating log
Connection from: 127.0.0.1 : 47352
New leader id: 345
Connection from: 127.0.0.1 : 47360
updating log

kali㉿kali:~/Desktop$ python3 client.py
No .env found, proceeding as normal
Frontend Environmental Address found: 3.227.253.129
Using default frontend HTTP port: 1234
Q: Query
B: Buy
O: Check Order
b
What product do you want?
while
How many?
2
Attempting to buy
{'data': {'order_number': 1903}}
Q: Query
B: Buy
O: Check Order
b
What product do you want?
Lulu
How many?
3
Attempting to buy
{'data': {'order_number': 1904}}
Q: Query
B: Buy
O: Check Order
o
What is the order number to check?
1903
Checking order number
{'data': {'number': 1903, 'name': 'whale', 'quantity': 2}}
Q: Query
B: Buy
O: Check Order

```

Restarting the downed order service (bottom left) to see that it successfully requests the logs

from the other replicas, see the other replicas (bottom middle and right) sending the missing logs:

The screenshot shows a Linux desktop environment with four terminal windows open, each displaying the output of a Python script. The terminals are arranged in a 2x2 grid.

- Top Left Terminal:** Displays the output of a script that monitors environmental ports and TCP connections. It includes logs for port 1234 (HTTP), port 6789 (TCP), and port 456 (environmental cache). It also handles leader election and service registration.
- Top Right Terminal:** Displays the output of a script that monitors environmental ports and TCP connections. It includes logs for port 1234 (HTTP), port 6789 (TCP), and port 456 (environmental cache). It also handles leader election and service registration.
- Bottom Left Terminal:** Displays the output of a script that acquires a lock and performs a task. It includes logs for port 1234 (HTTP) and port 456 (environmental cache). It also handles leader election and service registration.
- Bottom Right Terminal:** Displays the output of a script that exports ORDER\_SERVICE\_ID=0 and runs another script. It includes logs for port 1234 (HTTP), port 6789 (TCP), and port 456 (environmental cache). It also handles leader election and service registration.

The desktop interface includes a dock at the bottom with various application icons, and the system tray is visible on the right side of the screen.

Kill the two order services (bottom middle and right) that were up and serviced an order while the original leader (bottom left) was down. Then performing an order check on the client to see that restarting the original leader successfully received the logs it missed while down:

```

ubuntu@ip-172-31-73-21:~/applications
Activities Terminal Apr 26 18:25
ubuntu@ip-172-31-73-21:~/applications
Activities Terminal Apr 26 18:25
ubuntu@ip-172-31-73-21:~/applications
Activities Terminal Apr 26 18:25

```

Logs from the three services:

- Service 456 (Left):**

  - Choosing leader
  - 456 is healthy!
  - Starting tcp server for catalog invalidations
  - socket is listening
  - Starting Front End Server
  - 128.119.202.15 - - [26/Apr/2024 22:21:55] "POST /orders HTTP/1.1" 200
  - Connection from: 127.0.0.1 : 50778
  - 128.119.202.15 - - [26/Apr/2024 22:22:18] "POST /orders HTTP/1.1" 200
  - Service ID 456 is down.
  - choosing leader
  - Service ID 456 is down.
  - 345 is healthy!
  - Tried to alert Service ID of leader but 456 is down.
  - Connection from: 127.0.0.1 : 50800
  - 128.119.202.15 - - [26/Apr/2024 22:22:46] "GET /orders/1903 HTTP/1.1" 200
  - Service ID 345 is down.
  - choosing leader
  - 456 is healthy!
  - Tried to alert Service ID of leader but 345 is down.
  - Tried to alert Service ID of leader but 123 is down.

- Service 345 (Middle):**

  - Traceback (most recent call last):
 

```

File "/usr/lib/python3.6/threading.py", line 1294, in _shutdown
    n.join()
      File "/usr/lib/python3.6/threading.py", line 1056, in join
        self._wait_for_tstate_lock()
      File "/usr/lib/python3.6/threading.py", line 1072, in _wait_for_tstate_lock
        elif lock.acquire(block, timeout):
KeyboardInterrupt
      
```
  - Catalog Environmental Address Found: localhost
  - Catalog Environmental Port Found: 2319
  - Order Environmental Addresses Found: ['localhost', 'localhost', 'localhost']
  - Order Environmental Ports Found: [1115, 1116, 1117]
  - Order Environmental IDs Found: [123, 345, 456]
  - Order service ID position environmental found: 2
  - Received missing logs
  - Order socket binder to port 1117
  - socket is listening
  - Connection from: 127.0.0.1 : 34962
  - Connection from: 127.0.0.1 : 34964
  - Connection from: 127.0.0.1 : 34970
  - New leader id: 456

- Service 346 (Right):**

  - Traceback (most recent call last):
 

```

File "/usr/lib/python3.6/threading.py", line 1294, in _shutdown
    n.join()
      File "/usr/lib/python3.6/threading.py", line 1056, in join
        self._wait_for_tstate_lock()
      File "/usr/lib/python3.6/threading.py", line 1072, in _wait_for_tstate_lock
        elif lock.acquire(block, timeout):
KeyboardInterrupt
      
```
  - Catalog Environmental Address Found: localhost
  - Catalog Environmental Port Found: 2319
  - Order Environmental Addresses Found: ['localhost', 'localhost', 'localhost']
  - Order Environmental Ports Found: [1115, 1116, 1117]
  - Order Environmental IDs Found: [123, 345, 456]
  - Order service ID position environmental found: 2
  - Received missing logs
  - Order socket binder to port 1117
  - socket is listening
  - Connection from: 127.0.0.1 : 34962
  - Connection from: 127.0.0.1 : 34964
  - Connection from: 127.0.0.1 : 34970
  - New leader id: 456

```

kali㉿kali:~/Desktop
File Actions Edit View Help
(kali㉿kali:~/Desktop)
$ python3 client.py
No .env found, proceeding as normal
Frontend Environmental Address found: 3.227.253.129
Using default frontend HTTP port: 1234
Q: Query
B: Buy
O: Check Order
b
What product do you want?
whale
How many?
2
Attempting to buy
{'data': {'order_number': 1903}}
Q: Query
B: Buy
O: Check Order
b
What product do you want?
tux
How many?
3
Attempting to buy
{'data': {'order_number': 1904}}
Q: Query
B: Buy
O: Check Order
o
What is the order number to check?
1903
Checking order number
{'data': {'number': 1903, 'name': 'whale', 'quantity': 2}}
Q: Query
B: Buy
O: Check Order
o
What is the order number to check?
1904
Checking order number
{'data': {'number': 1904, 'name': 'tux', 'quantity': 3}}
Q: Query
B: Buy
O: Check Order

```

Looking at the order logs for each order service replica we can see that all have the same order logs:

The image shows a Ubuntu desktop environment with four terminal windows open, each displaying log files for a different service replica. The terminals are arranged in a 2x2 grid.

- Top Left Terminal:** Logs for service ID 456. It shows the service becoming leader, handling requests from 128.119.202.15, and attempting to alert other services about its leadership.
- Top Right Terminal:** Logs for service ID 345. It shows the service checking stock multiple times.
- Bottom Left Terminal:** Logs for service ID 193. It shows the service handling requests from 127.0.0.1 and 128.119.202.15.
- Bottom Right Terminal:** Logs for service ID 194. It shows the service handling requests from 127.0.0.1 and 128.119.202.15.

The logs in all four terminals are identical, indicating they are all running the same code and receiving the same external requests.

```
ubuntu@ip-172-31-73-21:~/applications$
```

```
ubuntu@ip-172-31-73-21:~/applications$
```

```
ubuntu@ip-172-31-73-21:~/applications$
```

```
ubuntu@ip-172-31-73-21:~/applications$
```