Indian Institute of Technology Jodhpur
Fundamentals of Distributed Systems

Assignment 1: Vector Clocks and Causal Ordering

Name: Saurabh Sharma
Roll No: <G24AI2060>
Submission Date: 25 June 2025

1. Objective

This project implements a **causally consistent key-value store** using **vector clocks**. It aims to capture the causal relationships between distributed events more accurately than Lamport clocks.

---

2. System Architecture

The system consists of 3 Dockerized nodes:

- Each node runs a Python Flask server (`node.py`)
- Nodes replicate writes to peers and use **vector clocks** to track causality
- Messages that arrive **out of order** are buffered until dependencies are met

```
vector-clock-kv-store/
├── src/
│   ├── node.py        ← main server logic
│   └── client.py     ← test client to verify causal consistency
├── Dockerfile
├── docker-compose.yml
└── project_report.pdf
```

---

3. Component Design

node.py:

- Hosts a Flask server
- Maintains a local key-value store and vector clock
- Implements causal delivery and buffering logic

client.py:

- Simulates a scenario:
    1. Write `x=5` to node1
    2. Read `x` from node2
    3. Update `x=10` on node3
- Tests causal consistency when messages are out of order

Dockerfile:

- Builds the Python app inside a container

docker-compose.yml:

- Launches 3 connected nodes (`node1`, `node2`, `node3`), each with a unique ID and shared network

---

4. Vector Clock Logic

Each node maintains a vector clock `{node1: x, node2: y, node3: z}`:

- Clock increments on each local write
- Merged when a message is received

- Write is **only applied** if:
    - `sender_clock[sender] == local_clock[sender] + 1`
    - All other entries `<= local_clock[entry]`

Otherwise, the message is **buffered** and applied later.

---

5. Test Scenario (client.py)

The client performs:

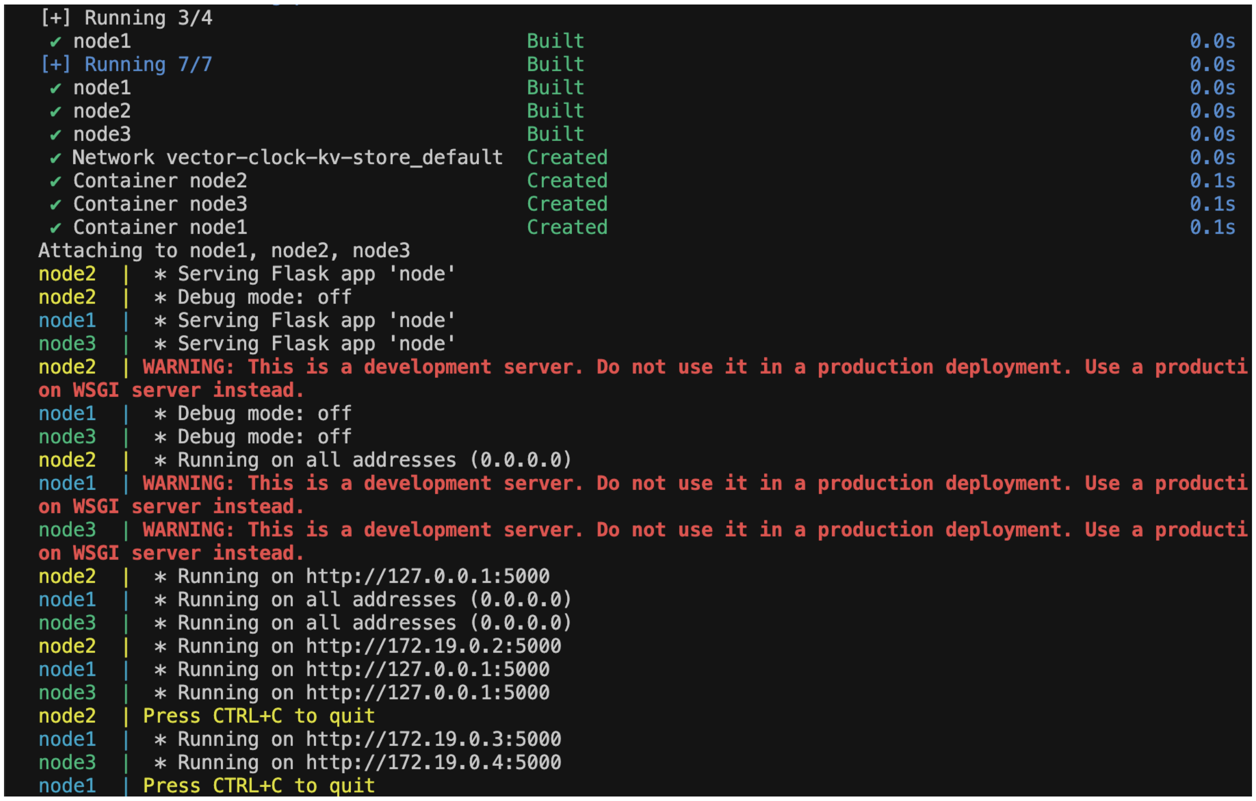
- Write `x=5` to node1
- Read `x` from node2
- Write `x=10` to node3
- Read final value from all nodes

Result:

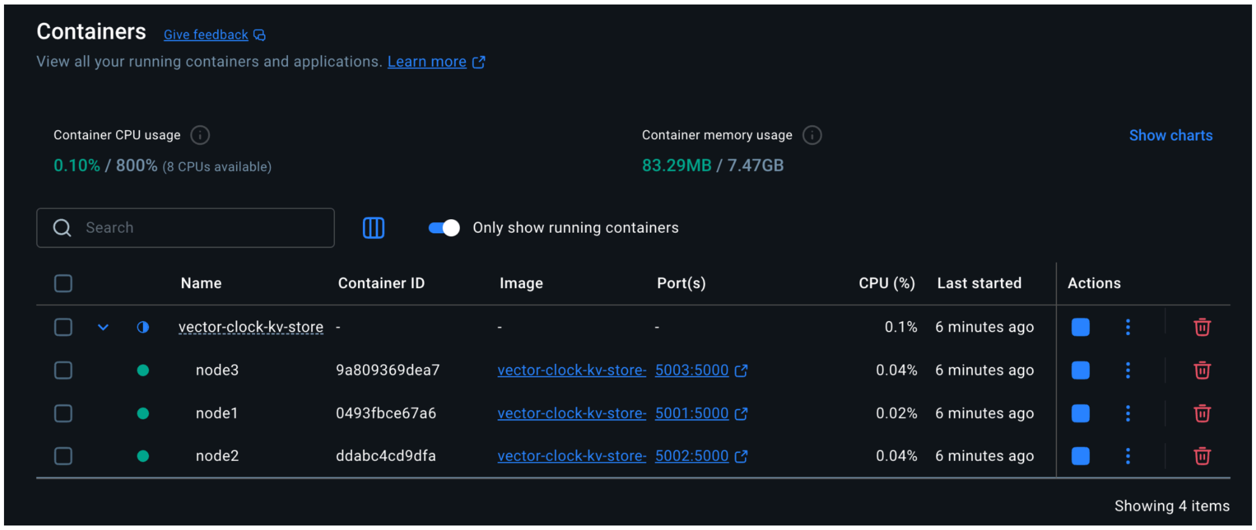
- All nodes eventually return `x=10`
- Vector clocks are updated and consistent
- Messages are only applied when causality is respected

---

6. Logs & Screenshots



```
[+] Running 3/4
 ✔ node1                                Built                                                    0.0s
[+] Running 7/7                         Built                                                    0.0s
 ✔ node1                                Built                                                    0.0s
 ✔ node2                                Built                                                    0.0s
 ✔ node3                                Built                                                    0.0s
 ✔ Network vector-clock-kv-store_default Created                                                  0.0s
 ✔ Container node2                      Created                                                  0.1s
 ✔ Container node3                      Created                                                  0.1s
 ✔ Container node1                      Created                                                  0.1s
Attaching to node1, node2, node3
node2  |  * Serving Flask app 'node'
node2  |  * Debug mode: off
node1  |  * Serving Flask app 'node'
node3  |  * Serving Flask app 'node'
node2  | WARNING: This is a development server. Do not use it in a production deployment. Use a producti
on WSGI server instead.
node1  |  * Debug mode: off
node3  |  * Debug mode: off
node2  |  * Running on all addresses (0.0.0.0)
node1  | WARNING: This is a development server. Do not use it in a production deployment. Use a producti
on WSGI server instead.
node3  | WARNING: This is a development server. Do not use it in a production deployment. Use a producti
on WSGI server instead.
node2  |  * Running on http://127.0.0.1:5000
node1  |  * Running on all addresses (0.0.0.0)
node3  |  * Running on all addresses (0.0.0.0)
node2  |  * Running on http://172.19.0.2:5000
node1  |  * Running on http://127.0.0.1:5000
node3  |  * Running on http://127.0.0.1:5000
node2  | Press CTRL+C to quit
node1  |  * Running on http://172.19.0.3:5000
node3  |  * Running on http://172.19.0.4:5000
node1  | Press CTRL+C to quit
```



Containers Give feedback

View all your running containers and applications. Learn more

Container CPU usage
0.10% / 800% (8 CPUs available)

Container memory usage
83.29MB / 7.47GB

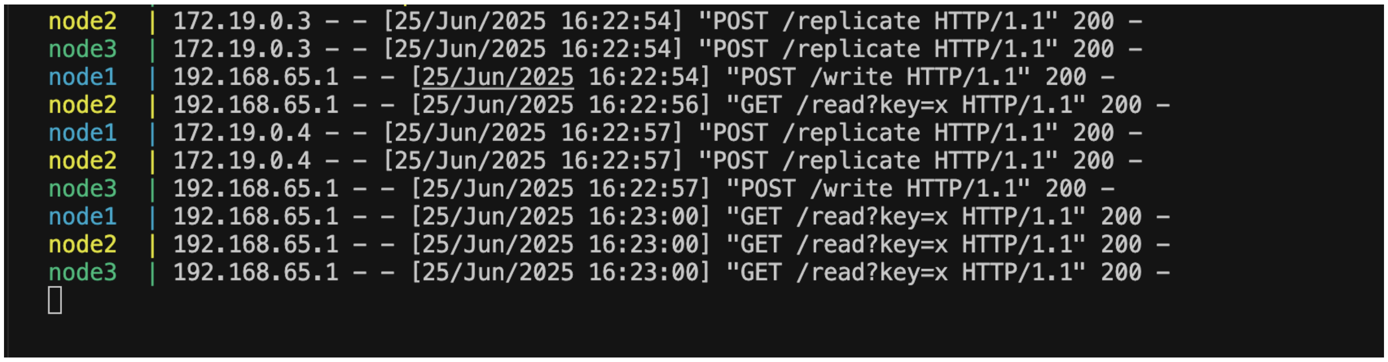
Show charts

Search

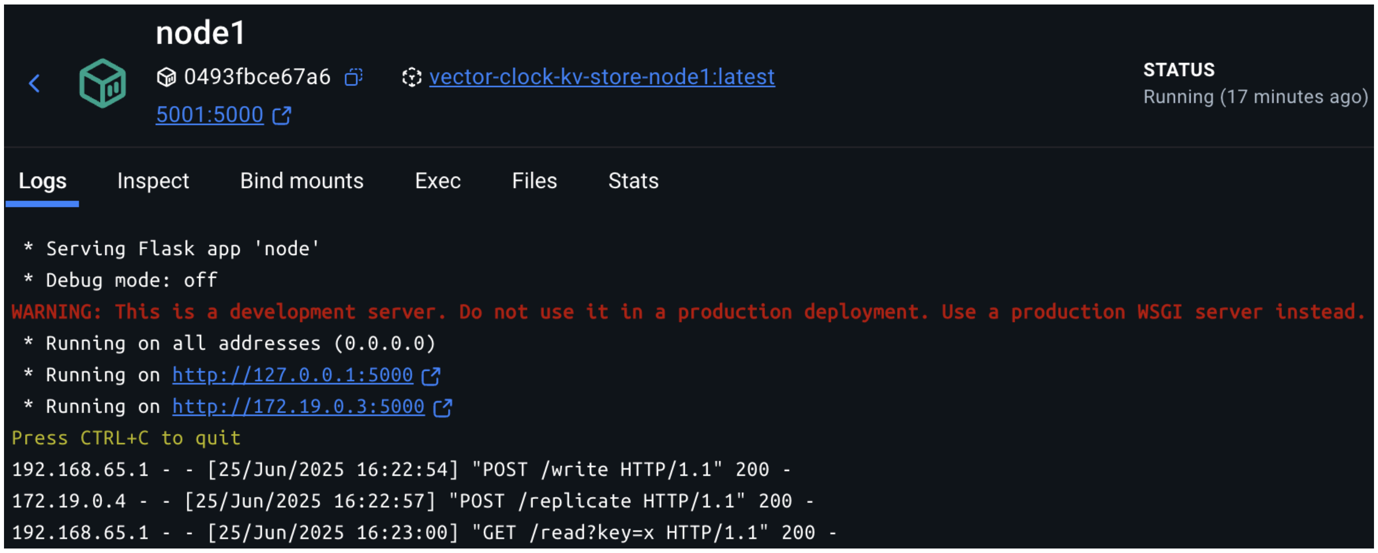
Only show running containers

	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
	vector-clock-kv-store	-	-	-	0.1%	6 minutes ago	
●	node3	9a809369dea7	vector-clock-kv-store-	5003:5000	0.04%	6 minutes ago	
●	node1	0493fbce67a6	vector-clock-kv-store-	5001:5000	0.02%	6 minutes ago	
●	node2	ddabc4cd9dfa	vector-clock-kv-store-	5002:5000	0.04%	6 minutes ago	

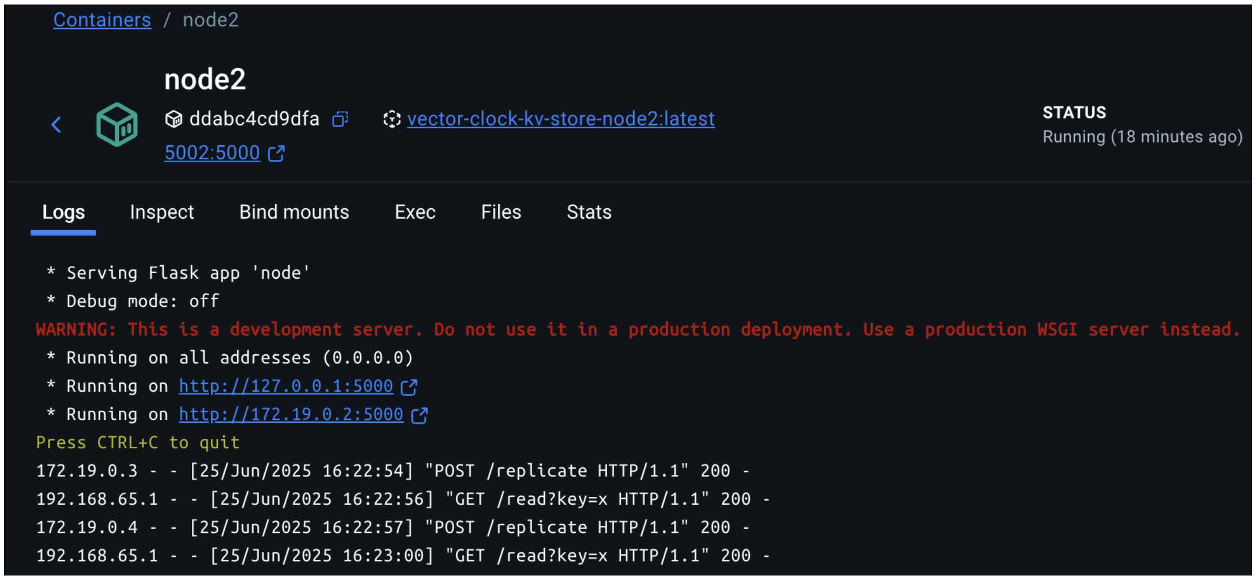
Showing 4 items



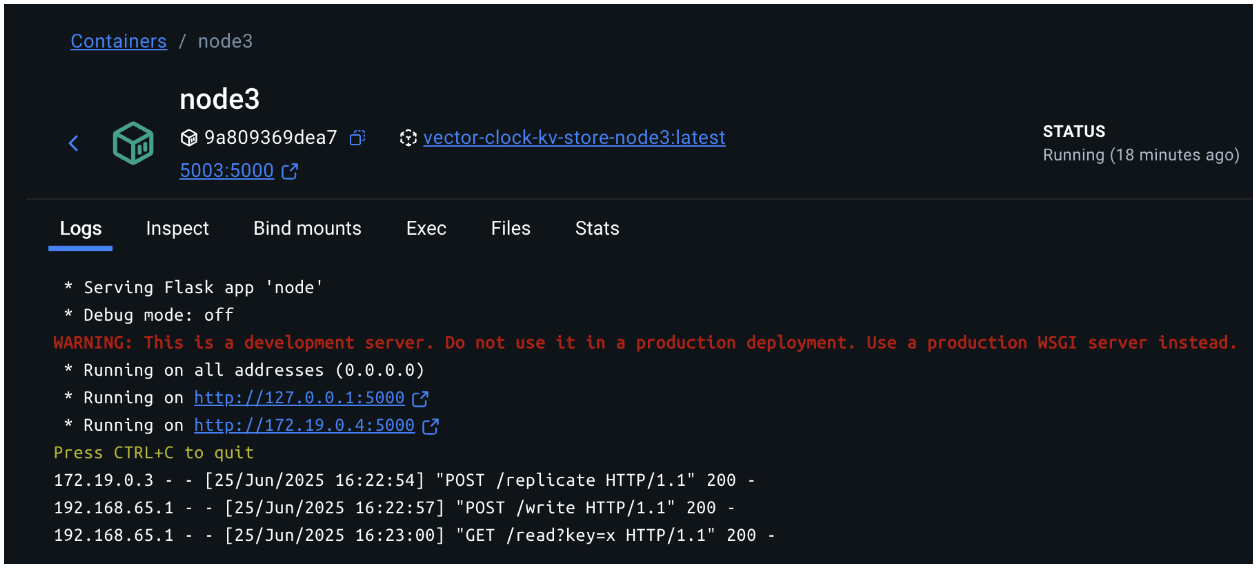
```
node2  | 172.19.0.3 - - [25/Jun/2025 16:22:54] "POST /replicate HTTP/1.1" 200 -
node3  | 172.19.0.3 - - [25/Jun/2025 16:22:54] "POST /replicate HTTP/1.1" 200 -
node1  | 192.168.65.1 - - [25/Jun/2025 16:22:54] "POST /write HTTP/1.1" 200 -
node2  | 192.168.65.1 - - [25/Jun/2025 16:22:56] "GET /read?key=x HTTP/1.1" 200 -
node1  | 172.19.0.4 - - [25/Jun/2025 16:22:57] "POST /replicate HTTP/1.1" 200 -
node2  | 172.19.0.4 - - [25/Jun/2025 16:22:57] "POST /replicate HTTP/1.1" 200 -
node3  | 192.168.65.1 - - [25/Jun/2025 16:22:57] "POST /write HTTP/1.1" 200 -
node1  | 192.168.65.1 - - [25/Jun/2025 16:23:00] "GET /read?key=x HTTP/1.1" 200 -
node2  | 192.168.65.1 - - [25/Jun/2025 16:23:00] "GET /read?key=x HTTP/1.1" 200 -
node3  | 192.168.65.1 - - [25/Jun/2025 16:23:00] "GET /read?key=x HTTP/1.1" 200 -
```



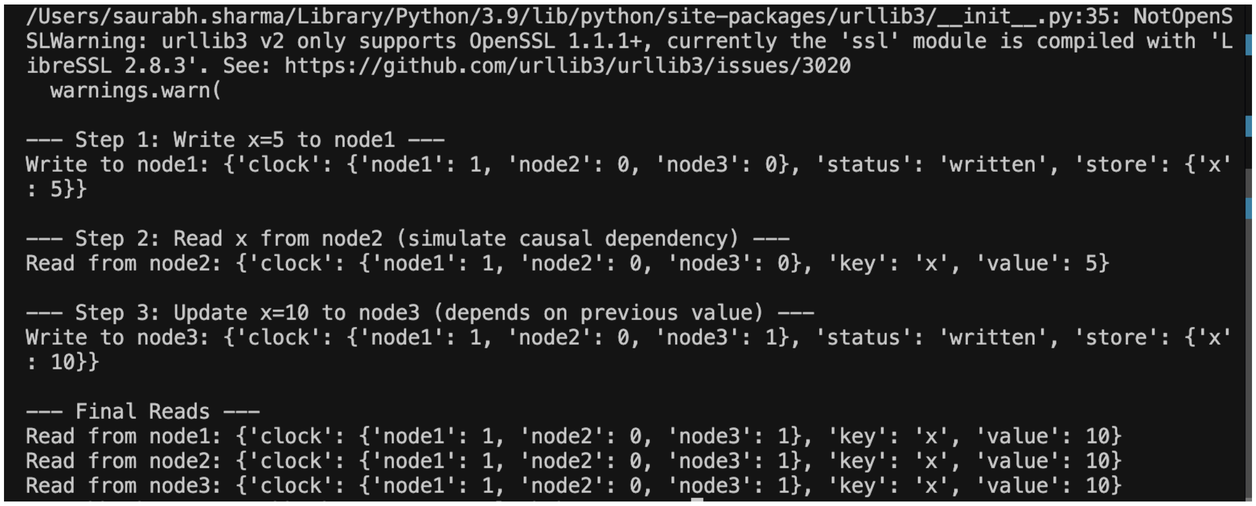
```
 * Serving Flask app 'node'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.19.0.3:5000
Press CTRL+C to quit
192.168.65.1 - - [25/Jun/2025 16:22:54] "POST /write HTTP/1.1" 200 -
172.19.0.4 - - [25/Jun/2025 16:22:57] "POST /replicate HTTP/1.1" 200 -
192.168.65.1 - - [25/Jun/2025 16:23:00] "GET /read?key=x HTTP/1.1" 200 -
```



```
 * Serving Flask app 'node'
 * Debug mode: off
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://172.19.0.2:5000
Press CTRL+C to quit
172.19.0.3 - - [25/Jun/2025 16:22:54] "POST /replicate HTTP/1.1" 200 -
192.168.65.1 - - [25/Jun/2025 16:22:56] "GET /read?key=x HTTP/1.1" 200 -
172.19.0.4 - - [25/Jun/2025 16:22:57] "POST /replicate HTTP/1.1" 200 -
192.168.65.1 - - [25/Jun/2025 16:23:00] "GET /read?key=x HTTP/1.1" 200 -
```



```
 * Serving Flask app 'node'
 * Debug mode: off
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://172.19.0.4:5000
Press CTRL+C to quit
172.19.0.3 - - [25/Jun/2025 16:22:54] "POST /replicate HTTP/1.1" 200 -
192.168.65.1 - - [25/Jun/2025 16:22:57] "POST /write HTTP/1.1" 200 -
192.168.65.1 - - [25/Jun/2025 16:23:00] "GET /read?key=x HTTP/1.1" 200 -
```



```
/Users/saurabh.sharma/Library/Python/3.9/lib/python/site-packages/urllib3/__init__.py:35: NotOpenS
SLWarning: urllib3 v2 only supports OpenSSL 1.1.1+, currently the 'ssl' module is compiled with 'L
ibreSSL 2.8.3'. See: https://github.com/urllib3/urllib3/issues/3020
  warnings.warn(

--- Step 1: Write x=5 to node1 ---
Write to node1: {'clock': {'node1': 1, 'node2': 0, 'node3': 0}, 'status': 'written', 'store': {'x'
: 5}}

--- Step 2: Read x from node2 (simulate causal dependency) ---
Read from node2: {'clock': {'node1': 1, 'node2': 0, 'node3': 0}, 'key': 'x', 'value': 5}

--- Step 3: Update x=10 to node3 (depends on previous value) ---
Write to node3: {'clock': {'node1': 1, 'node2': 0, 'node3': 1}, 'status': 'written', 'store': {'x'
: 10}}

--- Final Reads ---
Read from node1: {'clock': {'node1': 1, 'node2': 0, 'node3': 1}, 'key': 'x', 'value': 10}
Read from node2: {'clock': {'node1': 1, 'node2': 0, 'node3': 1}, 'key': 'x', 'value': 10}
Read from node3: {'clock': {'node1': 1, 'node2': 0, 'node3': 1}, 'key': 'x', 'value': 10}
```

---

7. Demo Video Link

---

8. Conclusion

This project demonstrated how vector clocks enable causal consistency across distributed nodes in a key-value store. All operations were successfully propagated in causal order, with vector clocks ensuring each node maintained a consistent and converged view of the data. While message buffering was not observed during this execution, the system is designed to delay and deliver updates only when their causal dependencies are met. Docker Compose was used to efficiently orchestrate multiple nodes and simulate a realistic distributed environment for testing.