Objective

The objective of this project is to implement a distributed key-value store that maintains causal consistency using Vector Clocks. Unlike Lamport clocks, vector clocks help identify and preserve causal relationships between events in a distributed system.

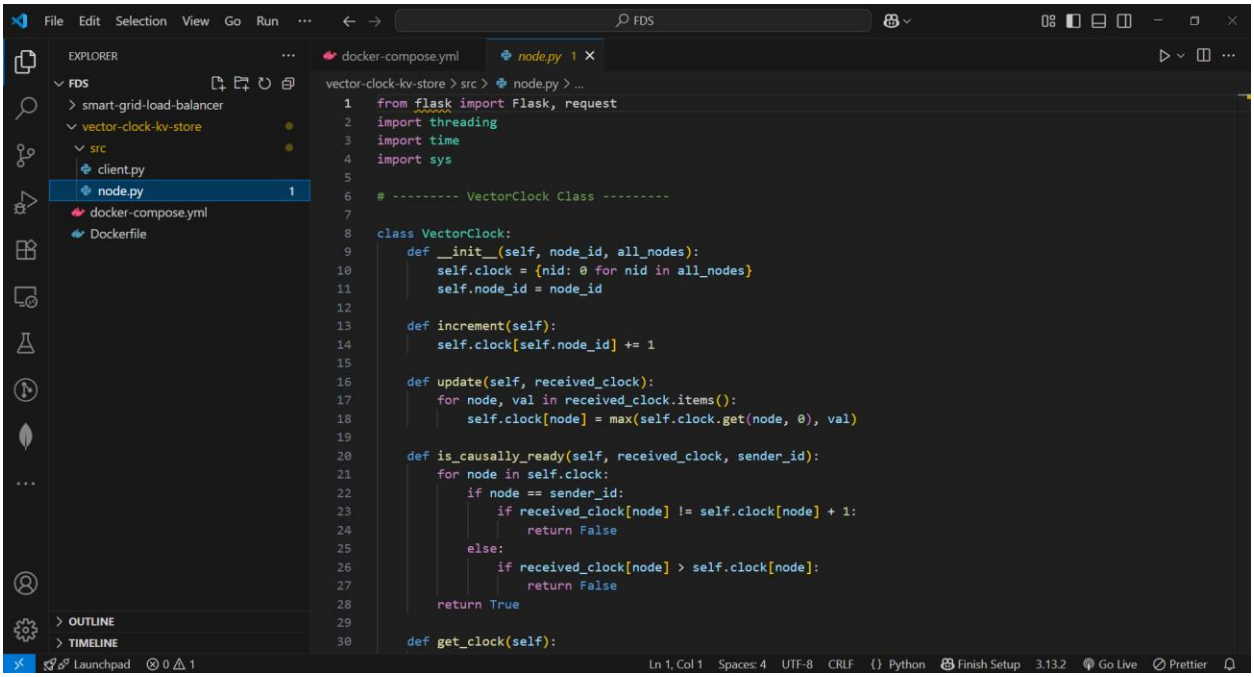
System Architecture

The system includes:

- **Three Flask-based Python nodes**
- **One client** to simulate PUT and GET operations
- **Docker and Docker Compose** for deployment and networking

Vector Clock Logic

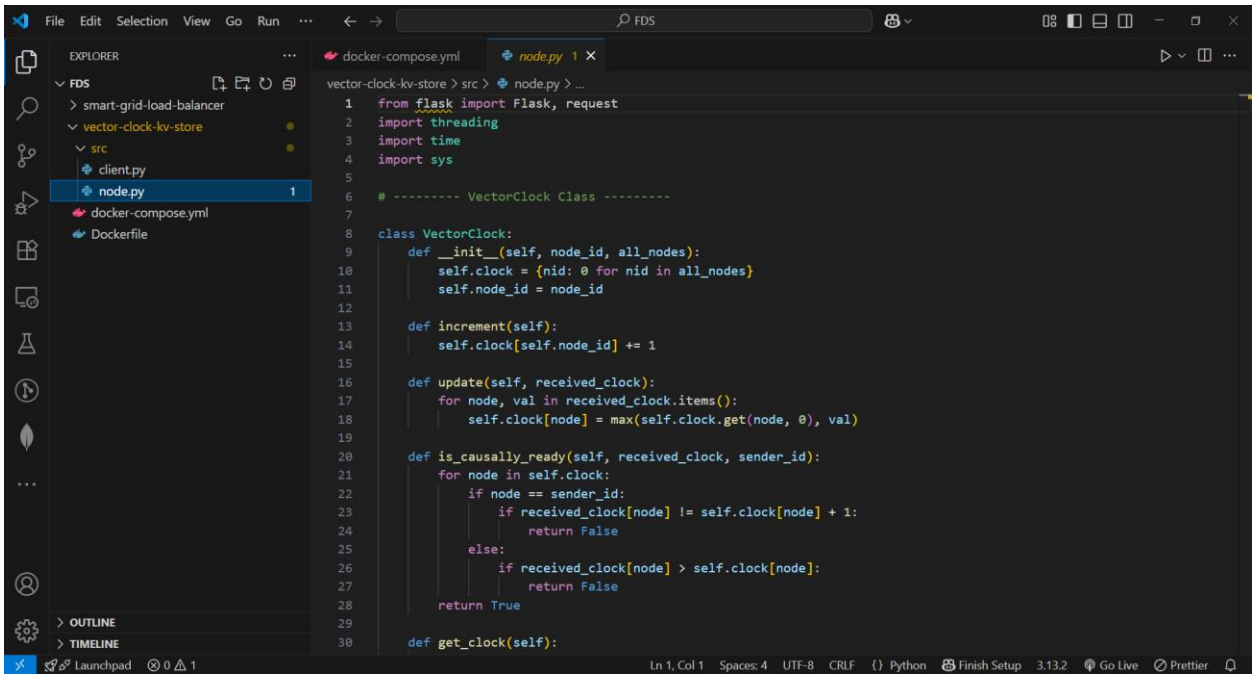
- Each node maintains a vector clock with entries for all nodes.
- **Increment:** On every local write operation.
- **Update:** On receiving a vector clock from another node.
- **Causally Ready:** A write is only applied if:
- For all i ≠ sender, msg_clock[i] ≤ local_clock[i]
- And msg_clock[sender] == local_clock[sender] + 1



Node Design (node.py)

Each node:

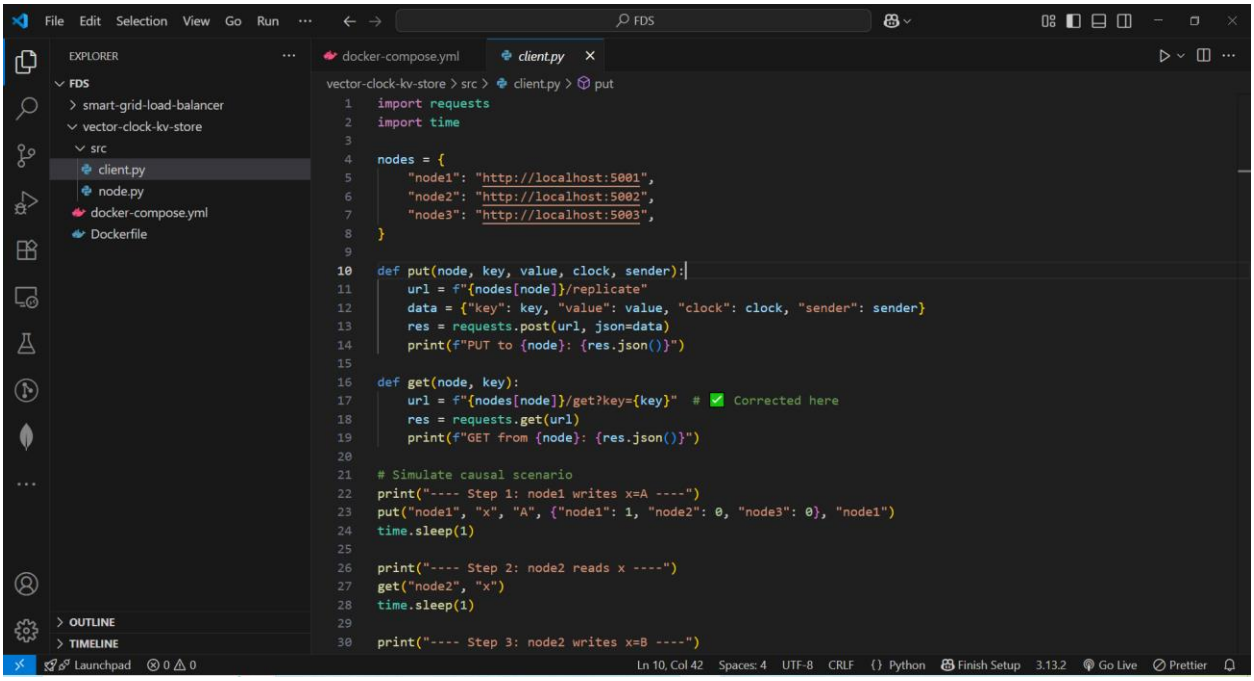
- Has a Flask server exposing:
    - GET /get?key=x
    - POST /replicate to receive updates
- Uses the VectorClock class to check causal readiness
- Buffers undeliverable messages until dependencies are resolved



Client Design (client.py)

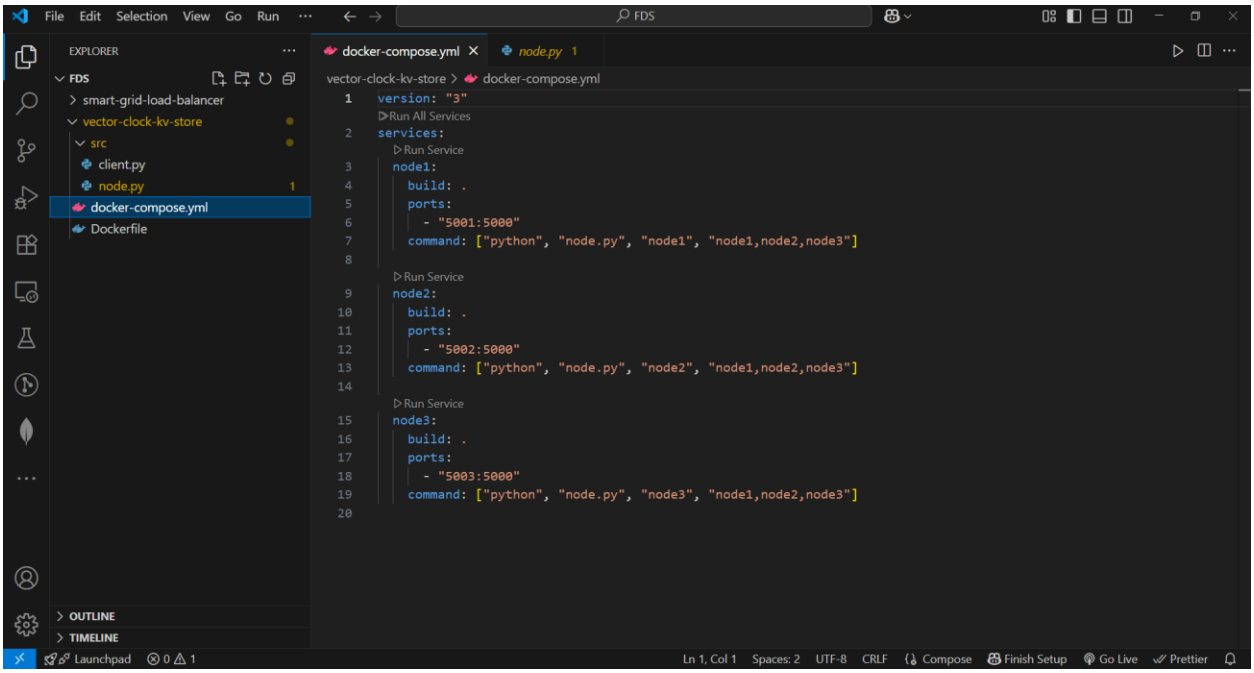
The client simulates the following scenario:

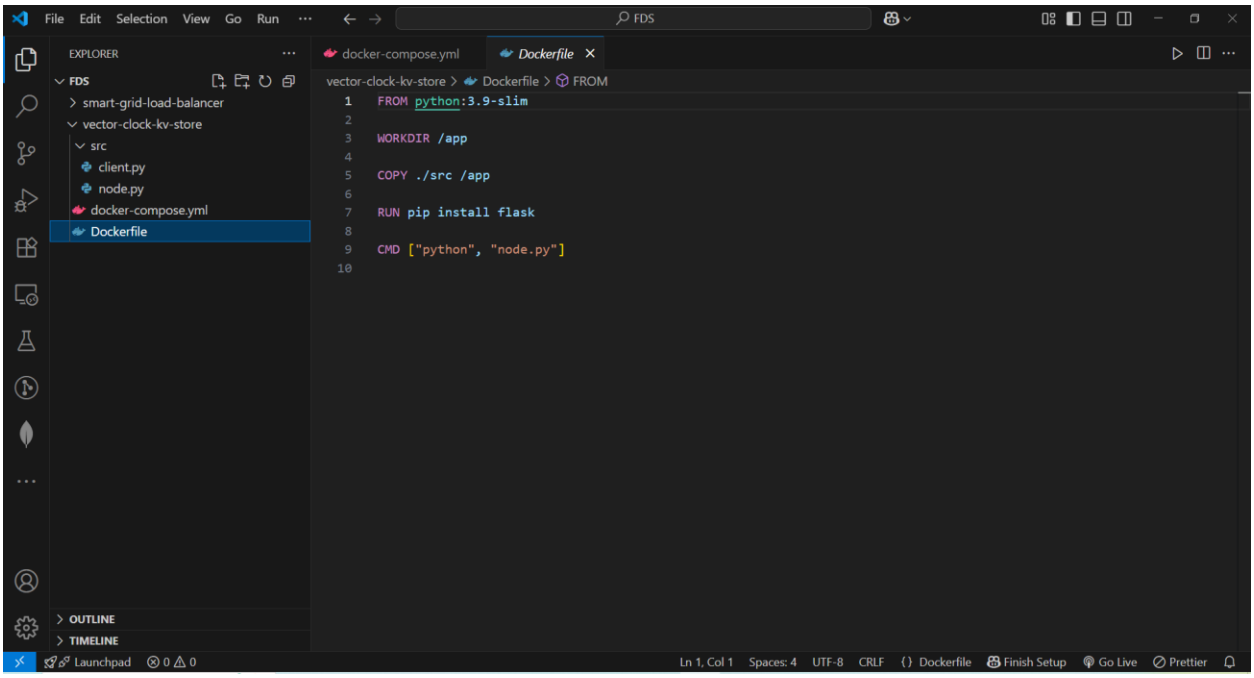
1. Step 1: node1 writes x = A
2. Step 2: node2 reads x
3. Step 3: node2 writes x = B, which is causally dependent on the previous read
4. Step 4: Ensure all nodes reflect consistent updates



Containerization

- Each node runs in a separate container.
- docker-compose.yml defines the services and ports.
- Dockerfile installs Flask and runs node.py

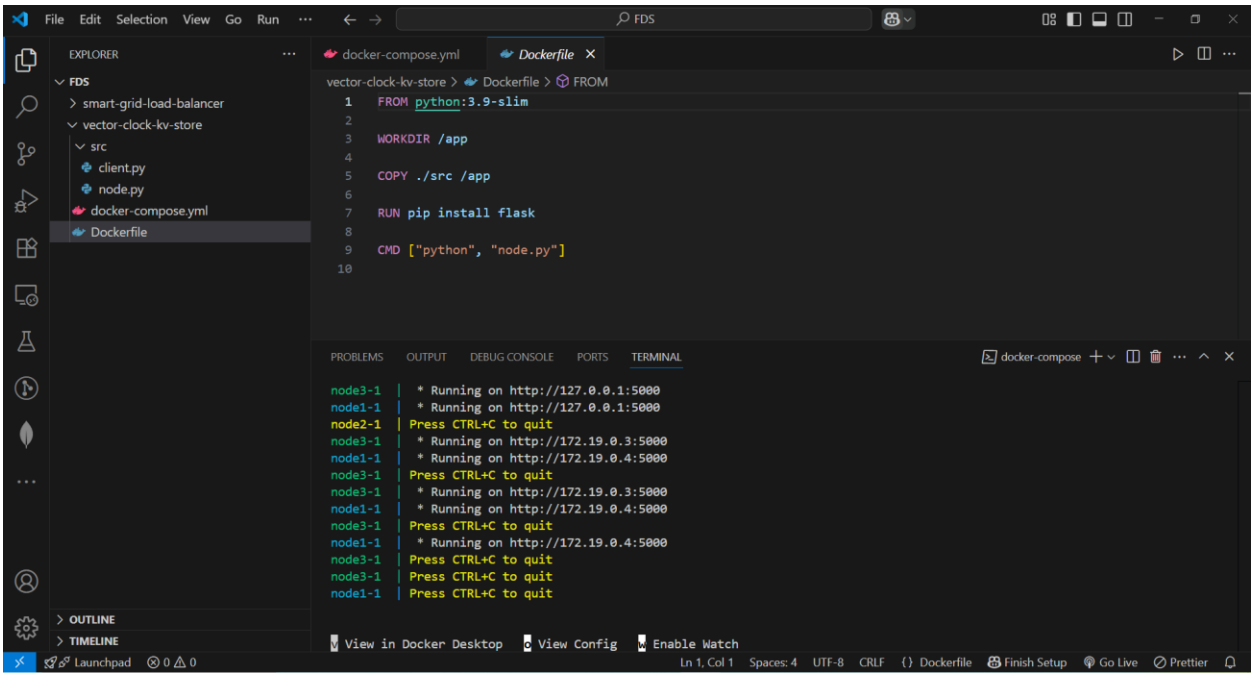




Testing and Output

Terminal Output

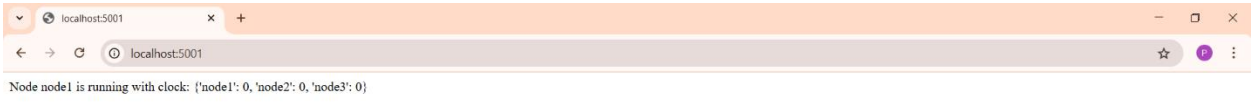
Logs confirm that all nodes start correctly and handle writes/reads as per causal rules.



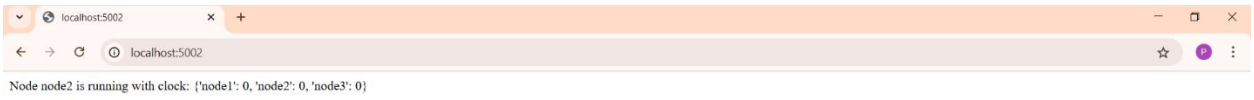
Node Views

Each node displays its status and vector clock correctly.

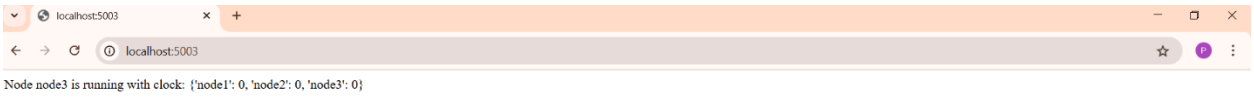
Node1



Node2


localhost:5002
localhost:5002
Node node2 is running with clock: {'node1': 0, 'node2': 0, 'node3': 0}

Node3


localhost:5003
localhost:5003
Node node3 is running with clock: {'node1': 0, 'node2': 0, 'node3': 0}

Conclusion

The system successfully maintains causal consistency using vector clocks. Buffered messages are only applied when their causal dependencies are met. Docker ensures the system is easily deployable and scalable.