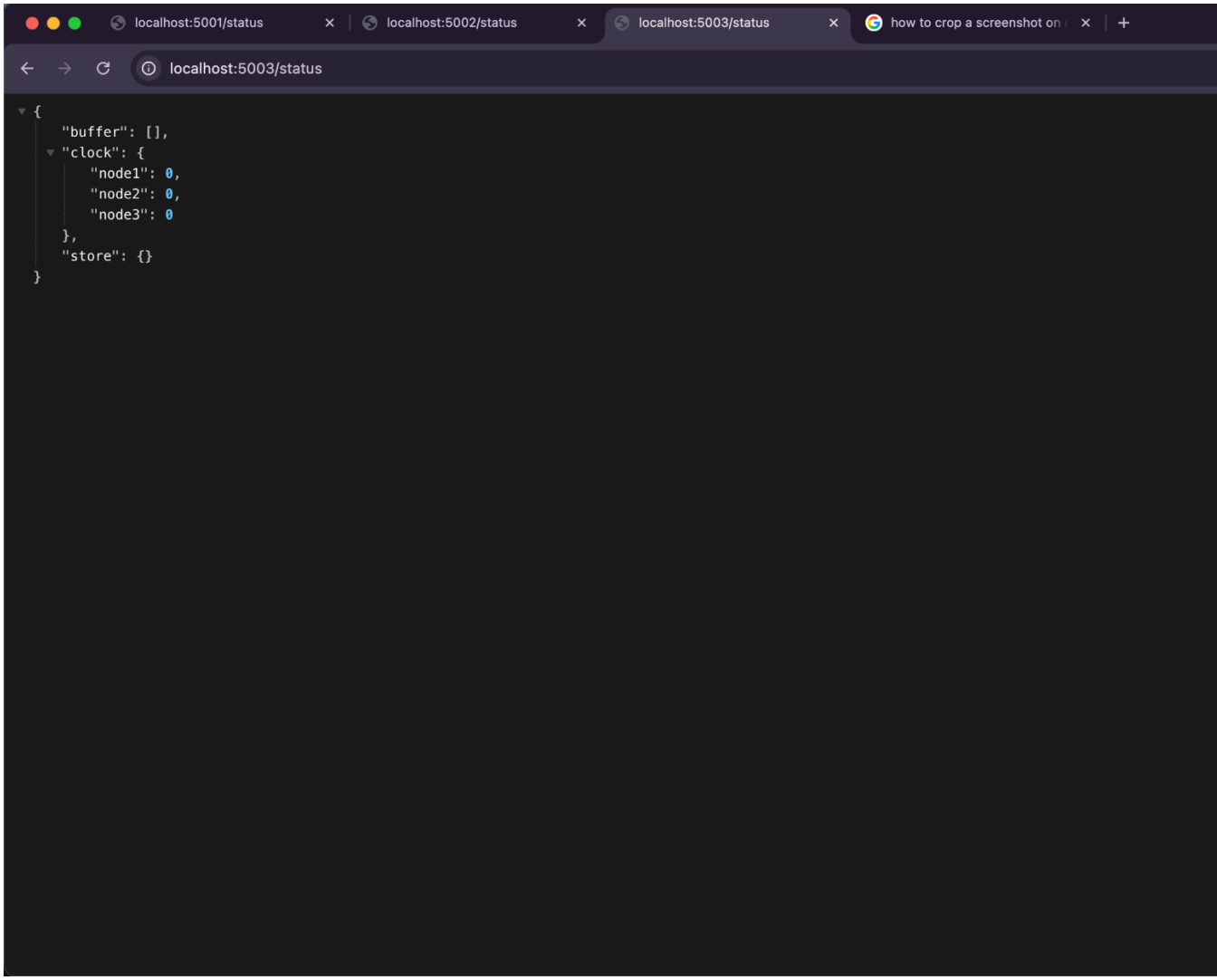
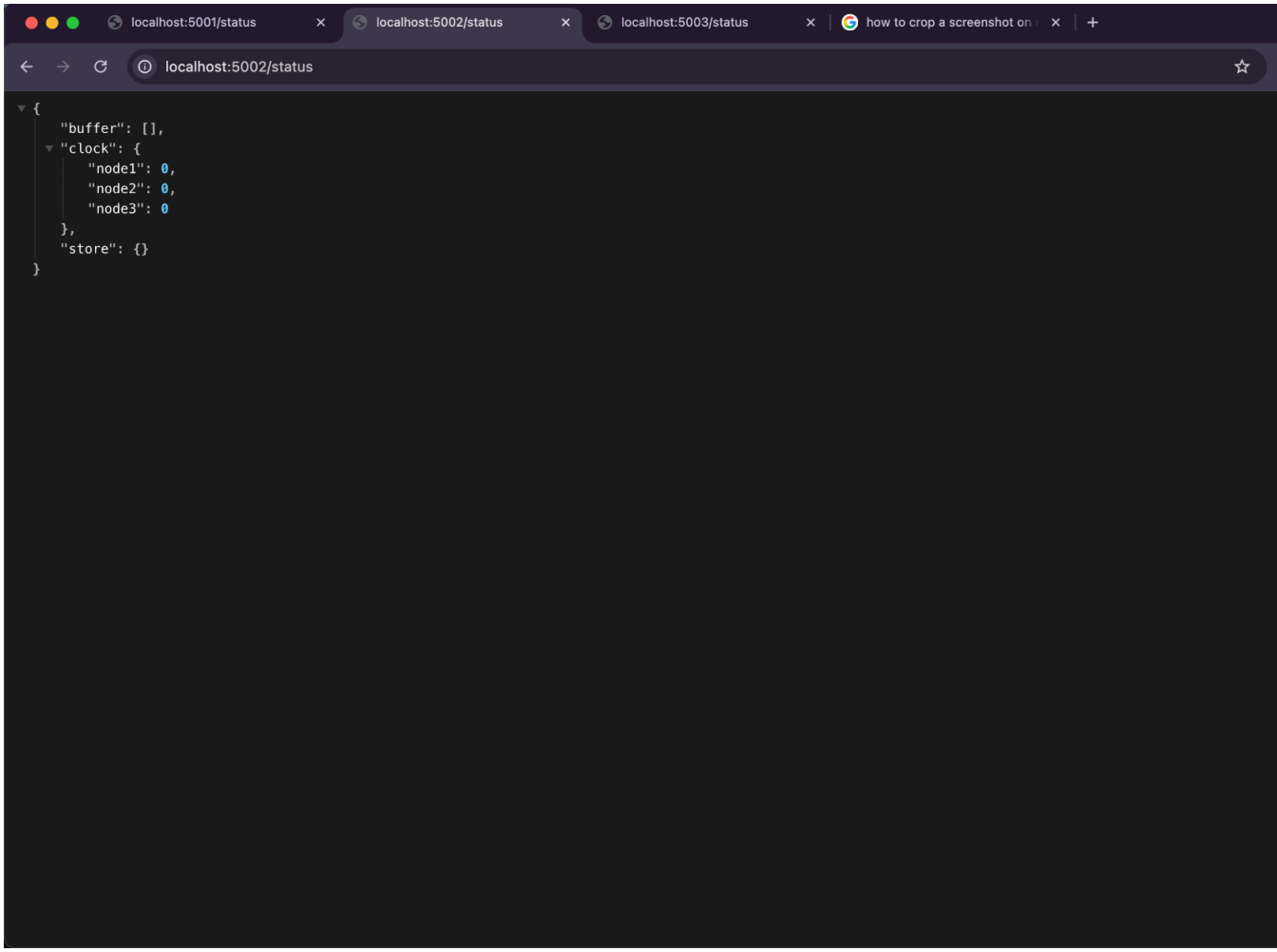
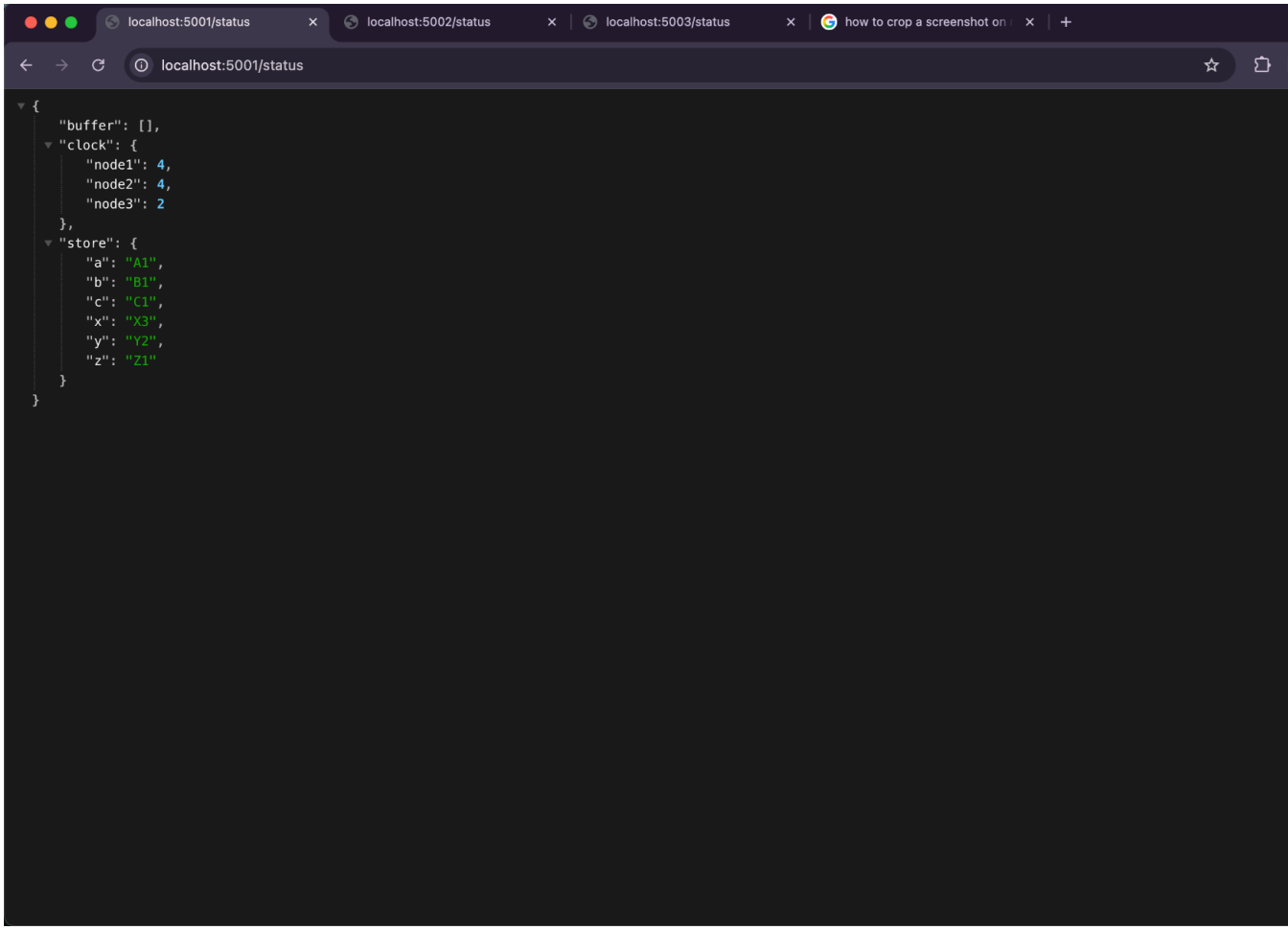
Yash Mathur
G24AI2006
FDS Assignment 1

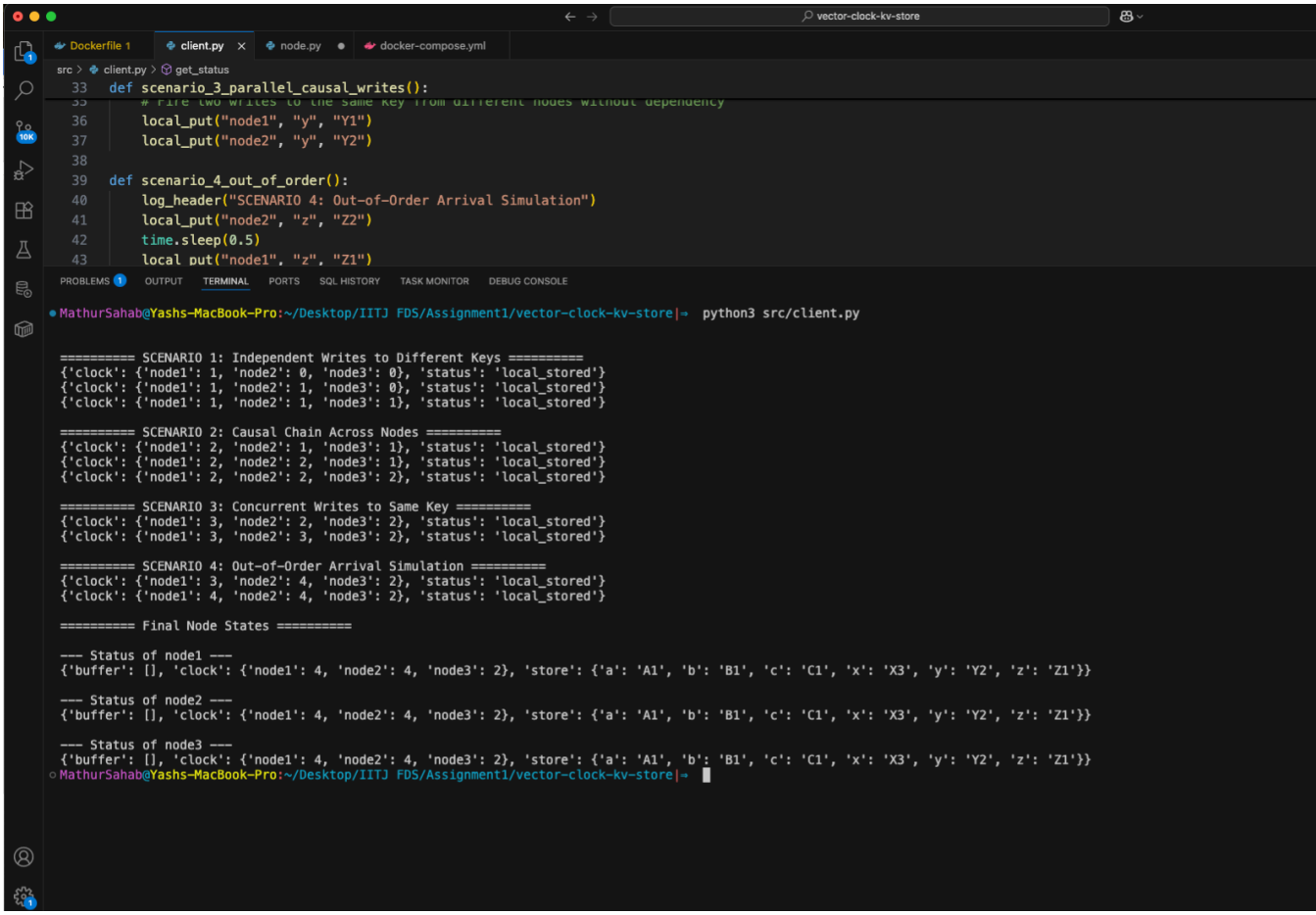
Architecture Summary

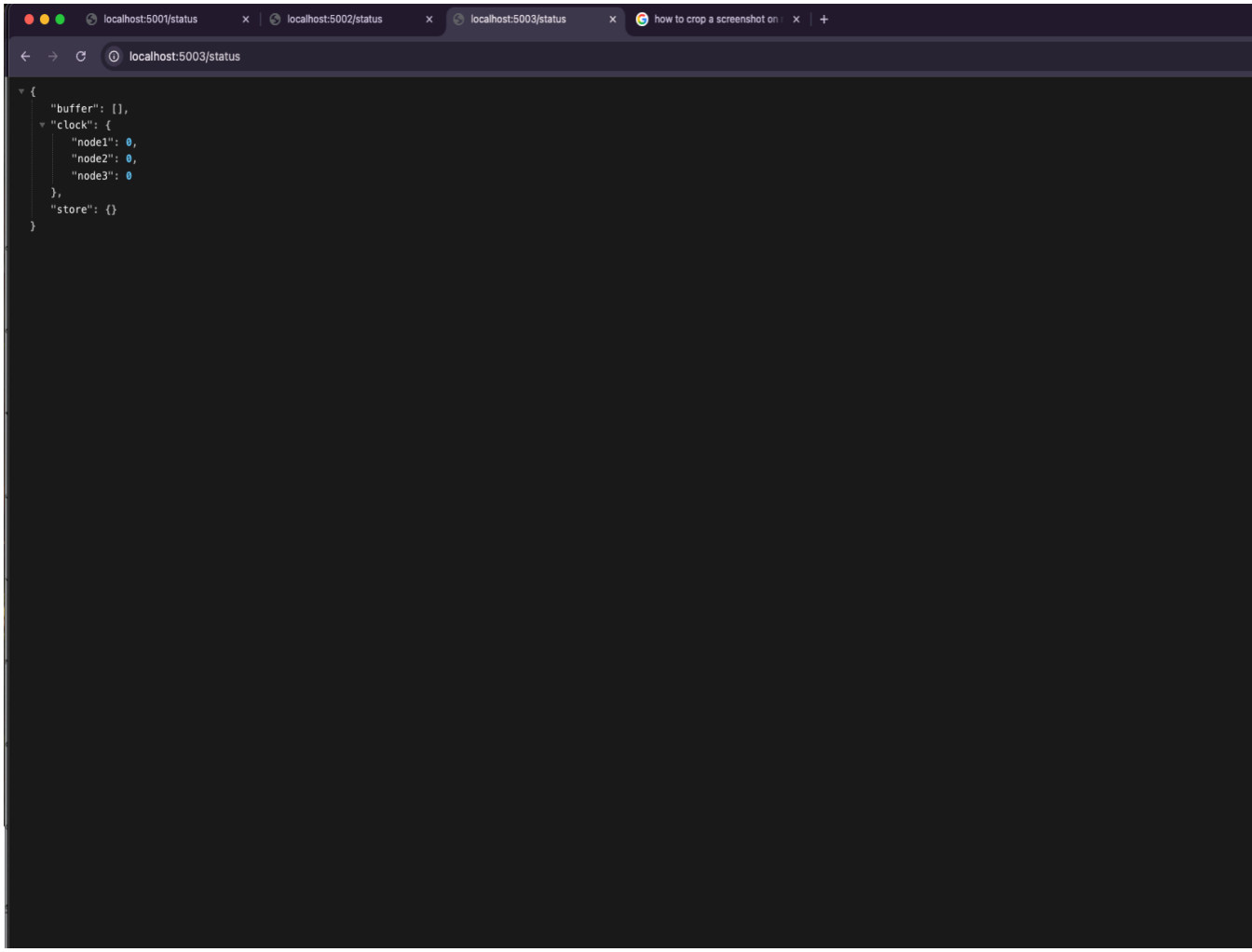
1. node.py key methods:
    a. Init_vector_clock: initializes all node values to 0.
    b. Increment_clock: increases vector clock value of current node by 1
    c. Update_clock: # obtain max value for each node and update in current node's vector clock
    d. Get_sender_from_vector: get sender node using the received vector clock
    e. Can_deliver: checks current node's vector clock value for every node, return false if even a single value is greater than existing value
    f. Deliver_buffered_messages: check and clears buffer values
    g. Put: this method receives the sent node's key-value pair and vector clock and evaluates whether to buffer
    h. Local_put: method to process client request locally; adds +1 to the value of current node's vector clock; updates key-value store
    i. Replicate_write: calls put method; provides current node's key-value pair and vector clock to other nodes
    j. Status: get method to output current status of each node
2. Docker-compose.yml:
    a. Nodes' names: node1, node2, node3
    b. Attached to Host Ports 5001, 5002, 5003
3. Dockerfile: Docker image to define containers. Runs command to download Flask, runs node.py
4. client.py key methods:
    a. Local_put: initiates post request and provides key-value pair.
    b. Scenario_1_independent_writes: independent Writes to Different Keys
    c. Scenario_2_causal_chain: Causal Chain Across Nodes
    d. Scenario_4_out_of_order: Out-of-Order Arrival Simulation
    e. Show_all_statuses: show final states of all nodes

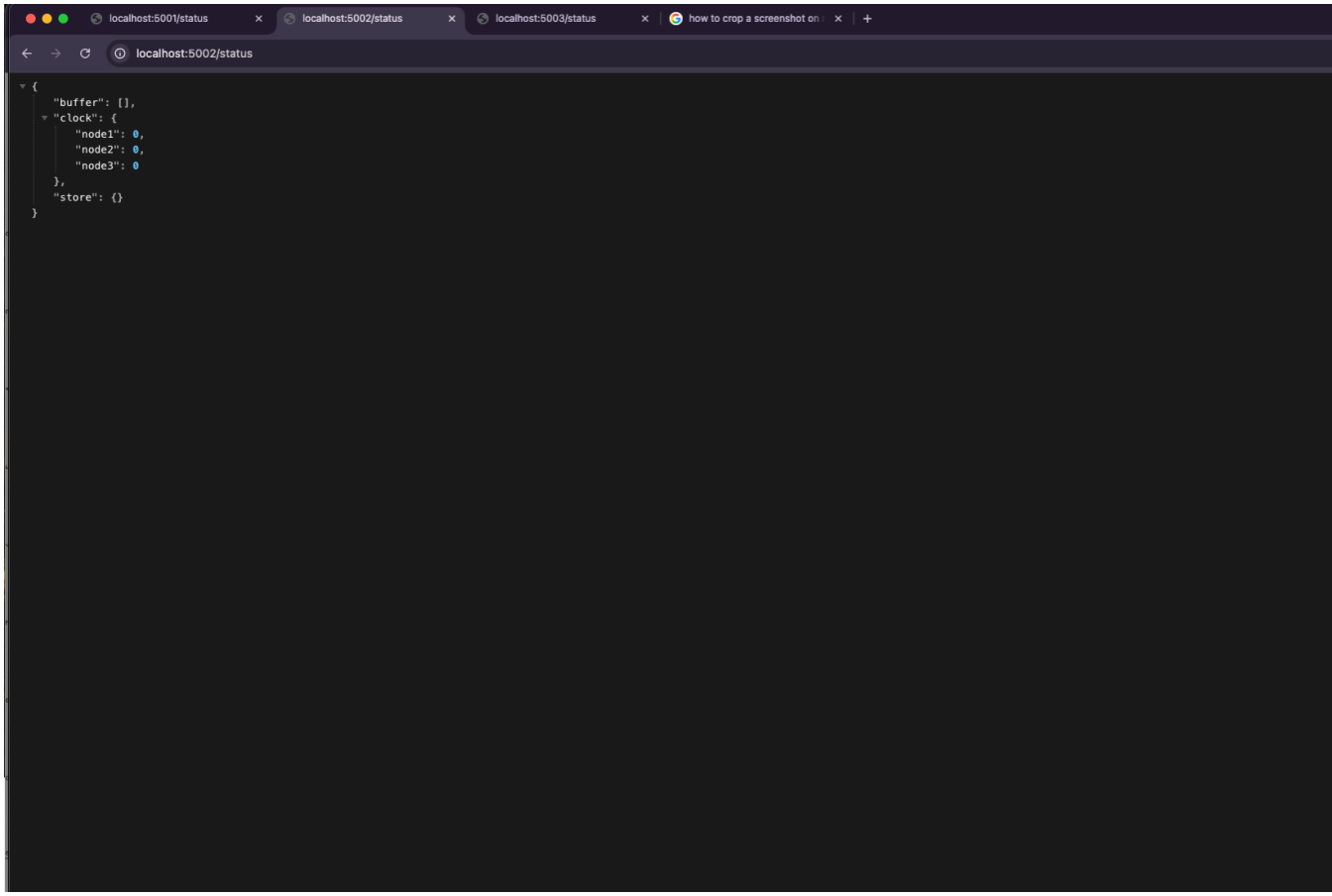

localhost:5001/status
localhost:5002/status
localhost:5003/status
how to crop a screenshot on
localhost:5003/status
{
"buffer": [],
"clock": {
"node1": 0,
"node2": 0,
"node3": 0
},
"store": {}
}

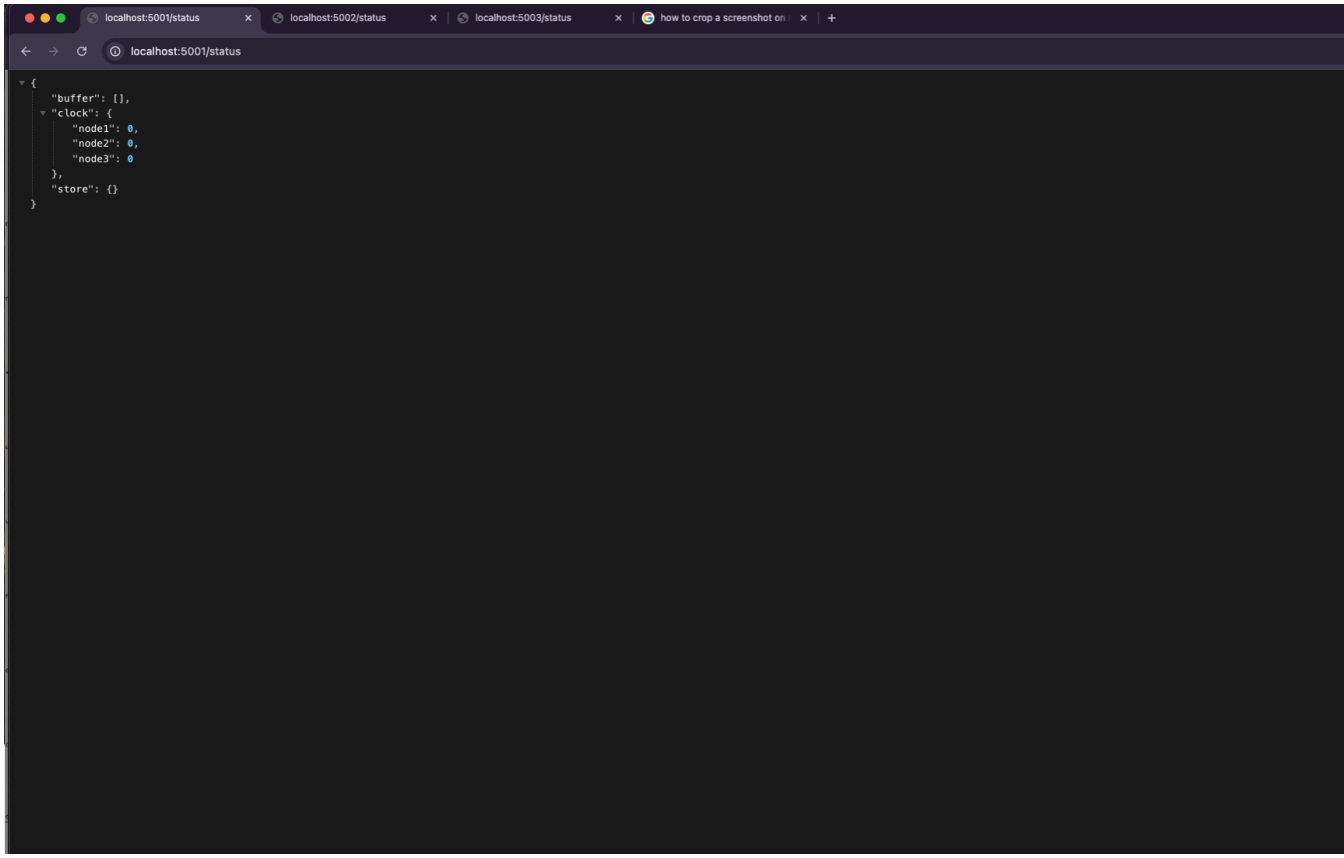

localhost:5001/status
localhost:5002/status
localhost:5003/status
how to crop a screenshot on
localhost:5002/status
{
"buffer": [],
"clock": {
"node1": 0,
"node2": 0,
"node3": 0
},
"store": {}
}

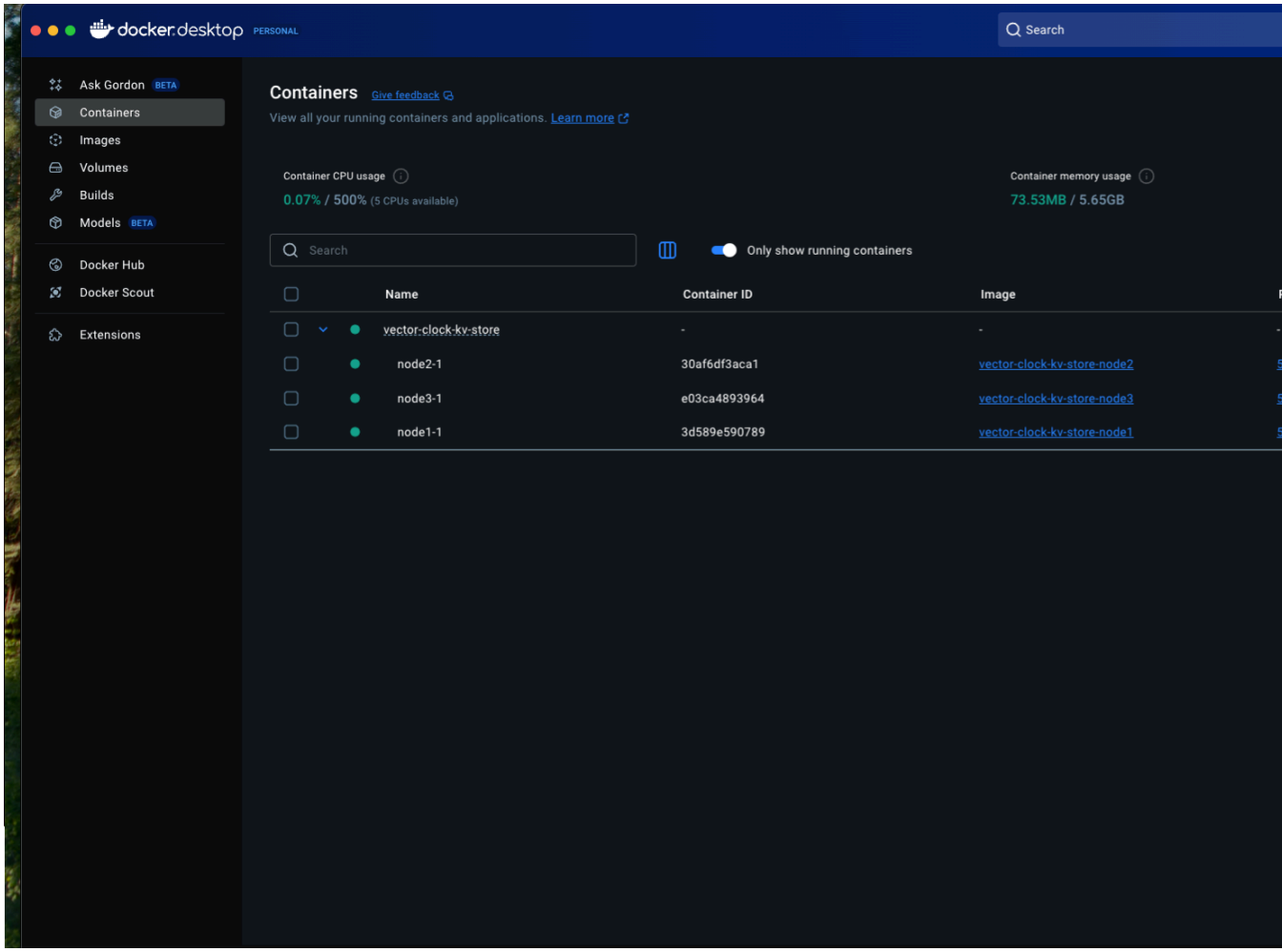

localhost:5001/status
localhost:5002/status
localhost:5003/status
how to crop a screenshot on
localhost:5001/status
{
"buffer": [],
"clock": {
"node1": 4,
"node2": 4,
"node3": 2
},
"store": {
"a": "A1",
"b": "B1",
"c": "C1",
"x": "X3",
"y": "Y2",
"z": "Z1"
}
}

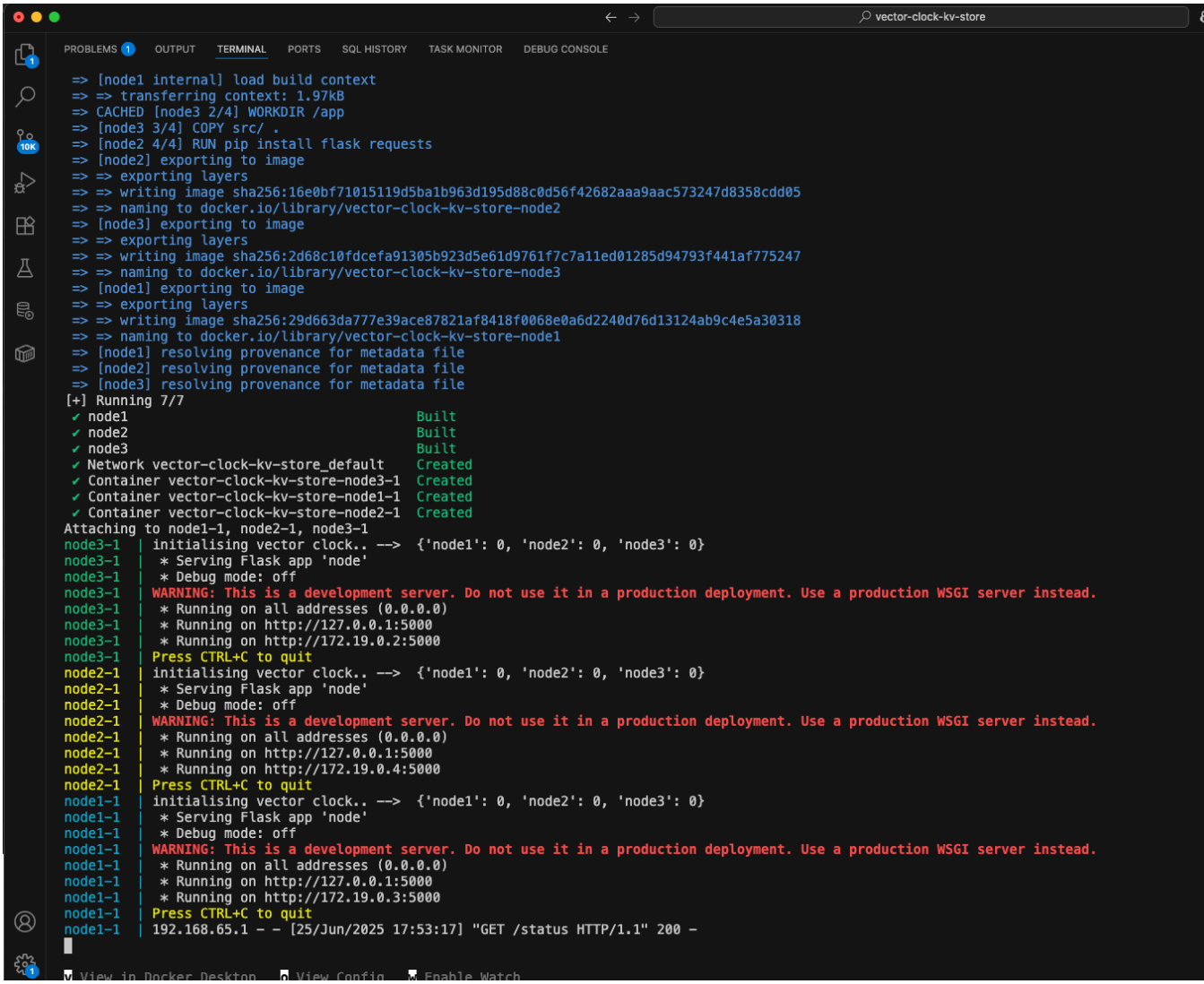

vector-clock-kv-store
Dockerfile 1
client.py
node.py
docker-compose.yml
src > client.py > get_status
def scenario_3_parallel_causal_writes():
# Fire two writes to the same key from different nodes without dependency
local_put("node1", "y", "Y1")
local_put("node2", "y", "Y2")
def scenario_4_out_of_order():
log_header("SCENARIO 4: Out-of-Order Arrival Simulation")
local_put("node2", "z", "Z2")
time.sleep(0.5)
local_put("node1", "z", "Z1")
PROBLEMS 1
OUTPUT
TERMINAL
PORTS
SQL HISTORY
TASK MONITOR
DEBUG CONSOLE
MathurSahab@Yashs-MacBook-Pro:~/Desktop/IITJ FDS/Assignment1/vector-clock-kv-store|⇒ python3 src/client.py
========== SCENARIO 1: Independent Writes to Different Keys ==========
{'clock': {'node1': 1, 'node2': 0, 'node3': 0}, 'status': 'local_stored'}
{'clock': {'node1': 1, 'node2': 1, 'node3': 0}, 'status': 'local_stored'}
{'clock': {'node1': 1, 'node2': 1, 'node3': 1}, 'status': 'local_stored'}
========== SCENARIO 2: Causal Chain Across Nodes ==========
{'clock': {'node1': 2, 'node2': 1, 'node3': 1}, 'status': 'local_stored'}
{'clock': {'node1': 2, 'node2': 2, 'node3': 1}, 'status': 'local_stored'}
{'clock': {'node1': 2, 'node2': 2, 'node3': 2}, 'status': 'local_stored'}
========== SCENARIO 3: Concurrent Writes to Same Key ==========
{'clock': {'node1': 3, 'node2': 2, 'node3': 2}, 'status': 'local_stored'}
{'clock': {'node1': 3, 'node2': 3, 'node3': 2}, 'status': 'local_stored'}
========== SCENARIO 4: Out-of-Order Arrival Simulation ==========
{'clock': {'node1': 3, 'node2': 4, 'node3': 2}, 'status': 'local_stored'}
{'clock': {'node1': 4, 'node2': 4, 'node3': 2}, 'status': 'local_stored'}
========== Final Node States ==========
--- Status of node1 ---
{'buffer': [], 'clock': {'node1': 4, 'node2': 4, 'node3': 2}, 'store': {'a': 'A1', 'b': 'B1', 'c': 'C1', 'x': 'X3', 'y': 'Y2', 'z': 'Z1'}}
--- Status of node2 ---
{'buffer': [], 'clock': {'node1': 4, 'node2': 4, 'node3': 2}, 'store': {'a': 'A1', 'b': 'B1', 'c': 'C1', 'x': 'X3', 'y': 'Y2', 'z': 'Z1'}}
--- Status of node3 ---
{'buffer': [], 'clock': {'node1': 4, 'node2': 4, 'node3': 2}, 'store': {'a': 'A1', 'b': 'B1', 'c': 'C1', 'x': 'X3', 'y': 'Y2', 'z': 'Z1'}}
MathurSahab@Yashs-MacBook-Pro:~/Desktop/IITJ FDS/Assignment1/vector-clock-kv-store|⇒


localhost:5001/status
localhost:5002/status
localhost:5003/status
how to crop a screenshot on
localhost:5003/status
{
"buffer": [],
"clock": {
"node1": 0,
"node2": 0,
"node3": 0
},
"store": {}
}


localhost:5001/status
localhost:5002/status
localhost:5003/status
how to crop a screenshot on
localhost:5002/status
{
"buffer": [],
"clock": {
"node1": 0,
"node2": 0,
"node3": 0
},
"store": {}
}


localhost:5001/status
localhost:5002/status
localhost:5003/status
how to crop a screenshot on
localhost:5001/status
{
"buffer": [],
"clock": {
"node1": 0,
"node2": 0,
"node3": 0
},
"store": {}
}


docker desktop
PERSONAL
Search
Ask Gordon BETA
Containers
Images
Volumes
Builds
Models BETA
Docker Hub
Docker Scout
Extensions
Containers
Give feedback
View all your running containers and applications. Learn more
Container CPU usage
0.07% / 500% (5 CPUs available)
Container memory usage
73.53MB / 5.65GB
Search
Only show running containers
Name
Container ID
Image
vector-clock-kv-store
-
-
node2-1
30af6df3aca1
vector-clock-kv-store-node2
node3-1
e03ca4893964
vector-clock-kv-store-node3
node1-1
3d589e590789
vector-clock-kv-store-node1


vector-clock-kv-store
PROBLEMS 1
OUTPUT
TERMINAL
PORTS
SQL HISTORY
TASK MONITOR
DEBUG CONSOLE
=> [node1 internal] load build context
=> => transferring context: 1.97kB
=> CACHED [node3 2/4] WORKDIR /app
=> [node3 3/4] COPY src/ .
=> [node2 4/4] RUN pip install flask requests
=> [node2] exporting to image
=> => exporting layers
=> => writing image sha256:16e0bf71015119d5ba1b963d195d88c0d56f42682aaa9aac573247d8358cdd05
=> => naming to docker.io/library/vector-clock-kv-store-node2
=> [node3] exporting to image
=> => exporting layers
=> => writing image sha256:2d68c10fdcefa91305b923d5e61d9761f7c7a11ed01285d94793f441af775247
=> => naming to docker.io/library/vector-clock-kv-store-node3
=> [node1] exporting to image
=> => exporting layers
=> => writing image sha256:29d663da777e39ace87821af8418f0068e0a6d2240d76d13124ab9c4e5a30318
=> => naming to docker.io/library/vector-clock-kv-store-node1
=> [node1] resolving provenance for metadata file
=> [node2] resolving provenance for metadata file
=> [node3] resolving provenance for metadata file
[+] Running 7/7
✔ node1 Built
✔ node2 Built
✔ node3 Built
✔ Network vector-clock-kv-store_default Created
✔ Container vector-clock-kv-store-node3-1 Created
✔ Container vector-clock-kv-store-node1-1 Created
✔ Container vector-clock-kv-store-node2-1 Created
Attaching to node1-1, node2-1, node3-1
node3-1 | initialising vector clock.. --> {'node1': 0, 'node2': 0, 'node3': 0}
node3-1 | * Serving Flask app 'node'
node3-1 | * Debug mode: off
node3-1 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
node3-1 | * Running on all addresses (0.0.0.0)
node3-1 | * Running on http://127.0.0.1:5000
node3-1 | * Running on http://172.19.0.2:5000
node3-1 | Press CTRL+C to quit
node2-1 | initialising vector clock.. --> {'node1': 0, 'node2': 0, 'node3': 0}
node2-1 | * Serving Flask app 'node'
node2-1 | * Debug mode: off
node2-1 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
node2-1 | * Running on all addresses (0.0.0.0)
node2-1 | * Running on http://127.0.0.1:5000
node2-1 | * Running on http://172.19.0.4:5000
node2-1 | Press CTRL+C to quit
node1-1 | initialising vector clock.. --> {'node1': 0, 'node2': 0, 'node3': 0}
node1-1 | * Serving Flask app 'node'
node1-1 | * Debug mode: off
node1-1 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
node1-1 | * Running on all addresses (0.0.0.0)
node1-1 | * Running on http://127.0.0.1:5000
node1-1 | * Running on http://172.19.0.3:5000
node1-1 | Press CTRL+C to quit
node1-1 | 192.168.65.1 - - [25/Jun/2025 17:53:17] "GET /status HTTP/1.1" 200 -
View in Docker Desktop
View Config
Enable Watch