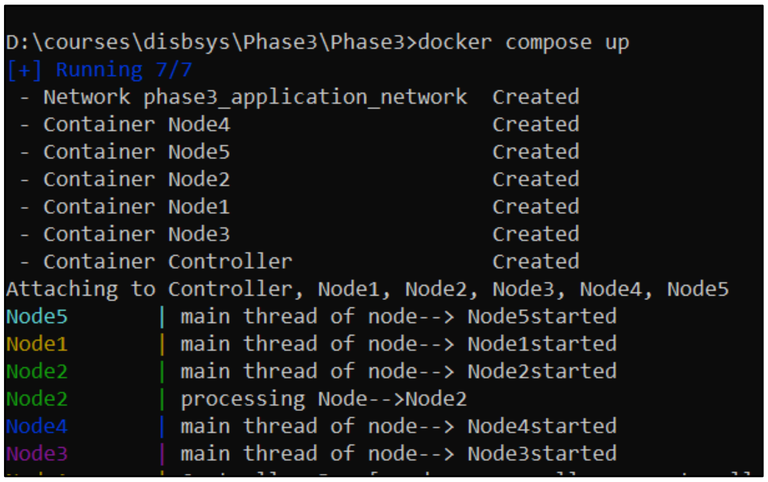
Validation of the RAFT Implementation

Along with all the images of node one controller image is also got created which is used to the validation of the developed algorithm.

Check the Demo video for more details

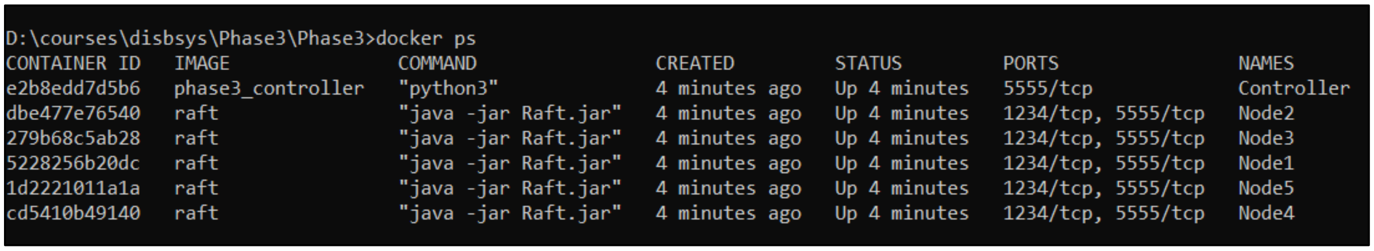
1. On running **$docker compose up** we get logs which shows that main thread of the 5 nodes get started and we also get some logs to see if all the containers execution is stable. Wait for the logging to get stopped.



```
D:\courses\disbsys\Phase3\Phase3>docker compose up
[+] Running 7/7
 - Network phase3_application_network  Created
 - Container Node4                     Created
 - Container Node5                     Created
 - Container Node2                     Created
 - Container Node1                     Created
 - Container Node3                     Created
 - Container Controller                Created
Attaching to Controller, Node1, Node2, Node3, Node4, Node5
Node5       | main thread of node--> Node5started
Node1       | main thread of node--> Node1started
Node2       | main thread of node--> Node2started
Node2       | processing Node-->Node2
Node4       | main thread of node--> Node4started
Node3       | main thread of node--> Node3started
```

Fig 4.1: logs after $docker compose up

2. **$ docker ps** shows the **running process, container name, image name and port** for all the five node containers and controller container



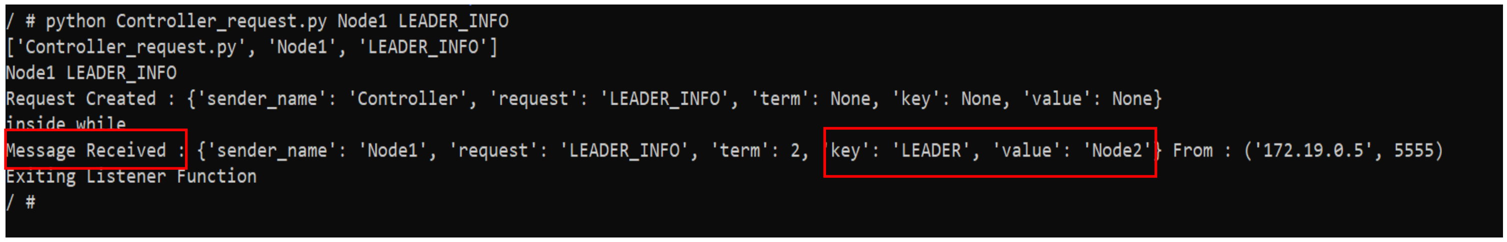
```
D:\courses\disbsys\Phase3\Phase3>docker ps
CONTAINER ID   IMAGE               COMMAND                CREATED         STATUS         PORTS                  NAMES
e2b8edd7d5b6   phase3_controller   "python3"              4 minutes ago   Up 4 minutes   5555/tcp               Controller
dbe477e76540   raft                "java -jar Raft.jar"   4 minutes ago   Up 4 minutes   1234/tcp, 5555/tcp     Node2
279b68c5ab28   raft                "java -jar Raft.jar"   4 minutes ago   Up 4 minutes   1234/tcp, 5555/tcp     Node3
5228256b20dc   raft                "java -jar Raft.jar"   4 minutes ago   Up 4 minutes   1234/tcp, 5555/tcp     Node1
1d2221011a1a   raft                "java -jar Raft.jar"   4 minutes ago   Up 4 minutes   1234/tcp, 5555/tcp     Node5
cd5410b49140   raft                "java -jar Raft.jar"   4 minutes ago   Up 4 minutes   1234/tcp, 5555/tcp     Node4
```

3. To test the system, we are provided with controller implementation in python. I modified the controller code to send **target and request type as the argument** while executing .py file.

   Examples of request command
   >>python Controller_request.py Node1 LEADER_INFO – This will send request to Node1 to request for LEADER_INFO. Node1 will return LEADER_INFO to the controller back which is printed as result in the Controller CLI.
   >>python Controller_request.py Node2 TIMEOUT – This will send request to Node2 to timeout immediately
   >>python Controller_request.py Node2 SHUTDOWN – This will send request to Node2 for shutting it down Node2 will exit without throwing any error.

   >>python Controller_request.py Node3 CONVERT_FOLLOWER – This will send request to Node3. If Node 3 is leader, then it will become follower and stop sending heartbeats.

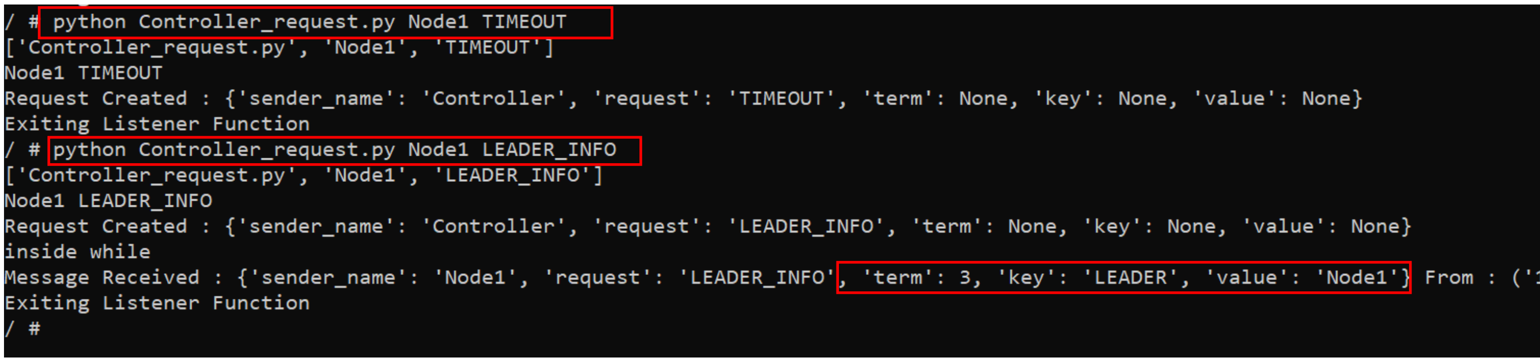
4. Open Controller CLI from docker windows and execute all the controller request
5. Request **LEADER_INFO –**
   The command **>>python Controller_request.py Node1 LEADER_INFO** gives the leader info provided by Node1



```
/ # python Controller_request.py Node1 LEADER_INFO
['Controller_request.py', 'Node1', 'LEADER_INFO']
Node1 LEADER_INFO
Request Created : {'sender_name': 'Controller', 'request': 'LEADER_INFO', 'term': None, 'key': None, 'value': None}
inside while
Message Received : {'sender_name': 'Node1', 'request': 'LEADER_INFO', 'term': 2, 'key': 'LEADER', 'value': 'Node2'} From : ('172.19.0.5', 5555)
Exiting Listener Function
/ #
```

We can see the message received from Node1 which shows **Leader is Node2**

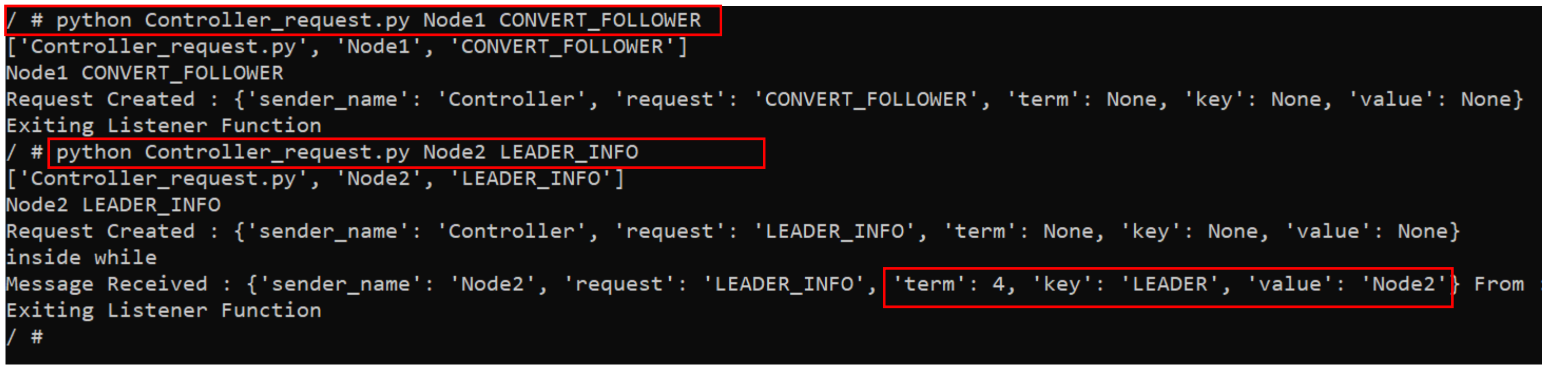
6. Request **TIMEOUT –**
   The command **>>python Controller_request.py Node1 TIMEOUT** will timeout the Node1 immediately and it will start election again with new term so a new leader will be selected.



```
/ # python Controller_request.py Node1 TIMEOUT
['Controller_request.py', 'Node1', 'TIMEOUT']
Node1 TIMEOUT
Request Created : {'sender_name': 'Controller', 'request': 'TIMEOUT', 'term': None, 'key': None, 'value': None}
Exiting Listener Function
/ # python Controller_request.py Node1 LEADER_INFO
['Controller_request.py', 'Node1', 'LEADER_INFO']
Node1 LEADER_INFO
Request Created : {'sender_name': 'Controller', 'request': 'LEADER_INFO', 'term': None, 'key': None, 'value': None}
inside while
Message Received : {'sender_name': 'Node1', 'request': 'LEADER_INFO', 'term': 3, 'key': 'LEADER', 'value': 'Node1'} From : ('
Exiting Listener Function
/ #
```

As seen in the screenshots after timeout of Node1, **Node1 becomes the leader in term=3** earlier Node2 was leader in term=2 which can be seen in the previous screenshots

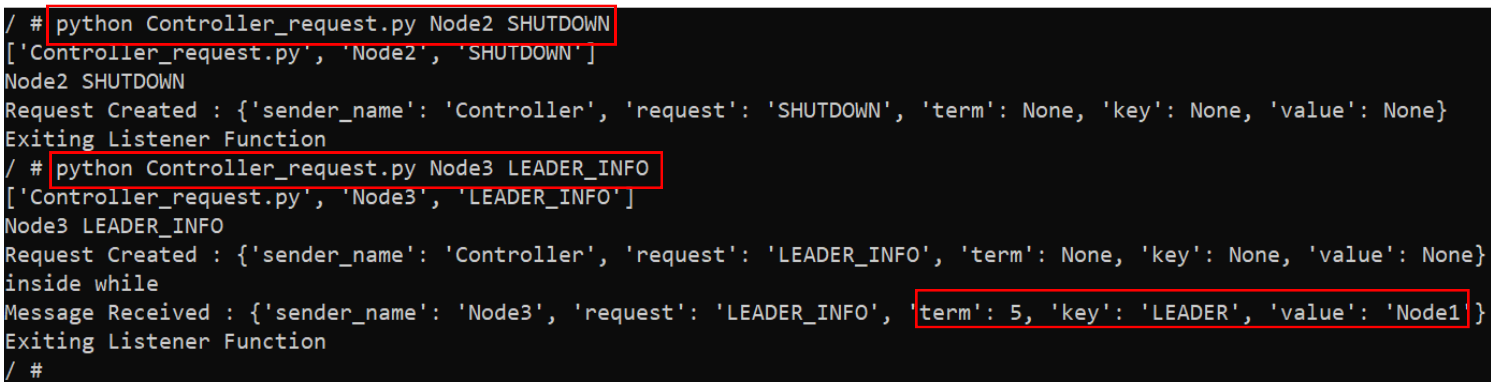
7. Request **CONVERT_FOLLOWER –**
   The command **>>python Controller_request.py Node1 CONVERT_FOLLOWER** will convert the Node1 to follower and a new leader will get selected.



```
/ # python Controller_request.py Node1 CONVERT_FOLLOWER
['Controller_request.py', 'Node1', 'CONVERT_FOLLOWER']
Node1 CONVERT_FOLLOWER
Request Created : {'sender_name': 'Controller', 'request': 'CONVERT_FOLLOWER', 'term': None, 'key': None, 'value': None}
Exiting Listener Function
/ # python Controller_request.py Node2 LEADER_INFO
['Controller_request.py', 'Node2', 'LEADER_INFO']
Node2 LEADER_INFO
Request Created : {'sender_name': 'Controller', 'request': 'LEADER_INFO', 'term': None, 'key': None, 'value': None}
inside while
Message Received : {'sender_name': 'Node2', 'request': 'LEADER_INFO', 'term': 4, 'key': 'LEADER', 'value': 'Node2'} From
Exiting Listener Function
/ #
```

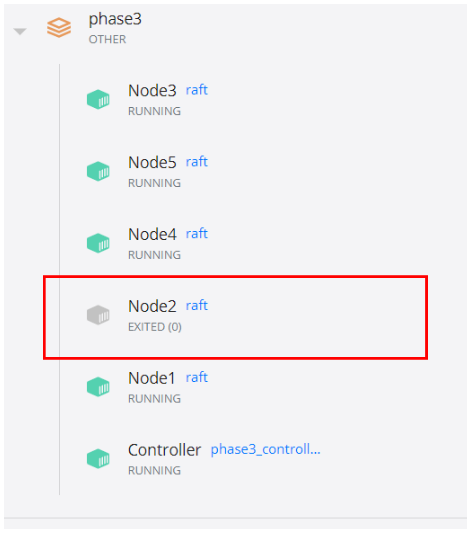
As seen in the screenshots after converting Node1 to the follower **Node2 becomes a leader in term=4**

8. Request **SHUTDOWN –**
   The command **>>python Controller_request.py Node2 SHUTDOWN** will shutdown the Node2 which was the leader in term=4. After shutdown new leader will be elected for term=5



```
/ # python Controller_request.py Node2 SHUTDOWN
['Controller_request.py', 'Node2', 'SHUTDOWN']
Node2 SHUTDOWN
Request Created : {'sender_name': 'Controller', 'request': 'SHUTDOWN', 'term': None, 'key': None, 'value': None}
Exiting Listener Function
/ # python Controller_request.py Node3 LEADER_INFO
['Controller_request.py', 'Node3', 'LEADER_INFO']
Node3 LEADER_INFO
Request Created : {'sender_name': 'Controller', 'request': 'LEADER_INFO', 'term': None, 'key': None, 'value': None}
inside while
Message Received : {'sender_name': 'Node3', 'request': 'LEADER_INFO', 'term': 5, 'key': 'LEADER', 'value': 'Node1'}
Exiting Listener Function
/ #
```

As seen in the screenshot new leader is selected in term=5 and below screenshot shows Node2 is stopped.



phase3
OTHER

Node3 raft
RUNNING

Node5 raft
RUNNING

Node4 raft
RUNNING

Node2 raft
EXITED (0)

Node1 raft
RUNNING

Controller phase3_controll...
RUNNING