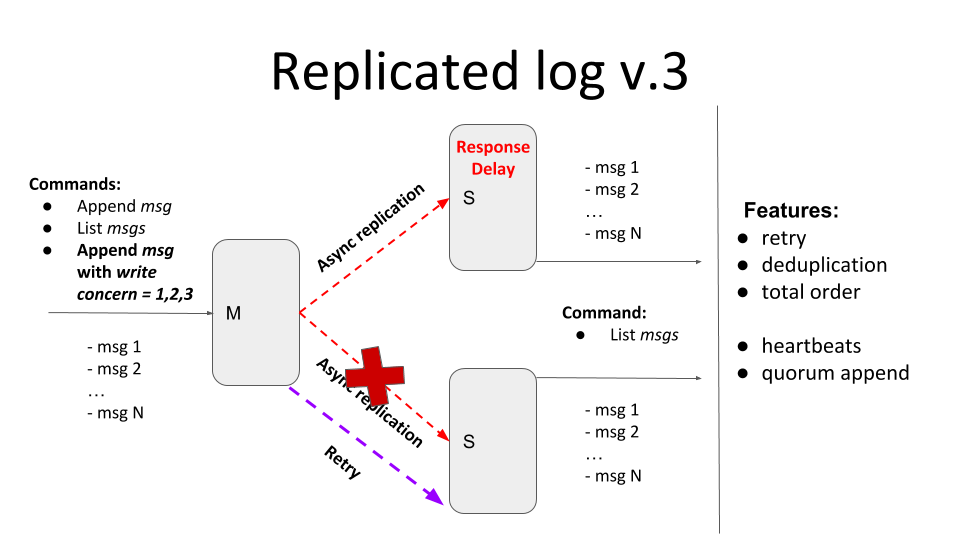
**Iteration 3.**

* **15 points**



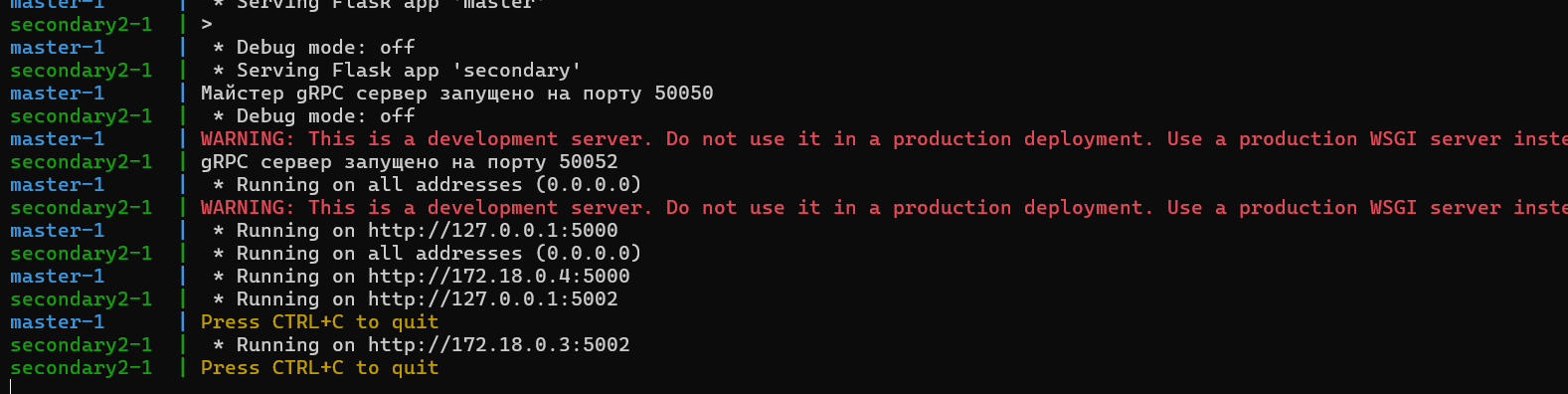
The current iteration should provide tunable semi-synchronicity for replication with a *retry* mechanism that should deliver all messages *exactly-once* in total order.

Main features:

* If message delivery fails (due to connection, or internal server error, or secondary is unavailable) the delivery attempts should be repeated - *retry*
  + If one of the secondaries is down and *w=3*, the client should be blocked until the node becomes available. Clients running in parallel shouldn’t be blocked by the blocked one.
  + If *w>1* the client should be blocked until the message will be delivered to all secondaries required by the write concern level. Clients running in parallel shouldn’t be blocked by the blocked one.
  + All messages that secondaries have missed due to unavailability should be replicated after (re)joining the master
  + Retries can be implemented with an unlimited number of attempts but, possibly, with some “smart” delays logic
  + You can specify a *timeout* for the master in case if there is no response from the secondary
* All messages should be present exactly once in the secondary log - *deduplication*
  + To test deduplication you can generate some random internal server error response from the secondary after the message has been added to the log
* The order of messages should be the same in all nodes - *total order*
  + If secondary has received messages *[msg1, msg2, msg4]*, it shouldn’t display the message *‘msg4’* until the *‘msg3’* will bereceived
  + To test the total order, you can generate some random internal server error response from the secondaries

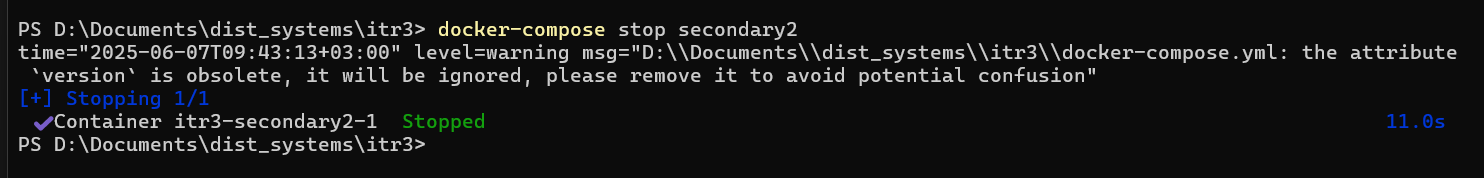
Self-check acceptance test:

1. Start M + S1
2. send (Msg1, W=1) - Ok
3. send (Msg2, W=2) - Ok
4. send (Msg3, W=3) - Wait
5. send (Msg4, W=1) - Ok
6. Start S2
7. Check messages on S2 - [Msg1, Msg2, Msg3, Msg4]

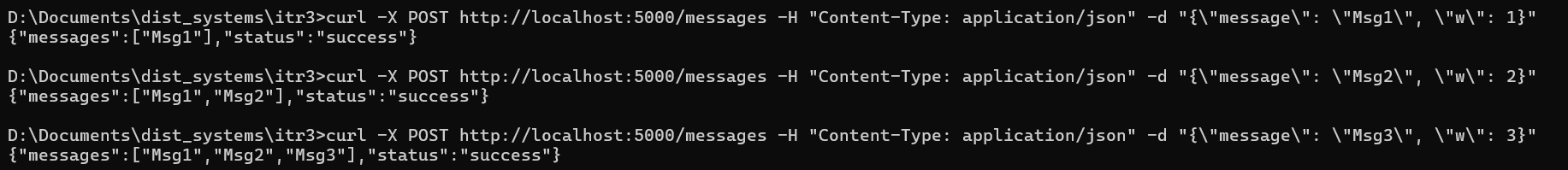




Зупиняємо:



Відправляємо повідомлення:



Пдініємаємо 2 секондарі:



Надсилаємо останній запит:

