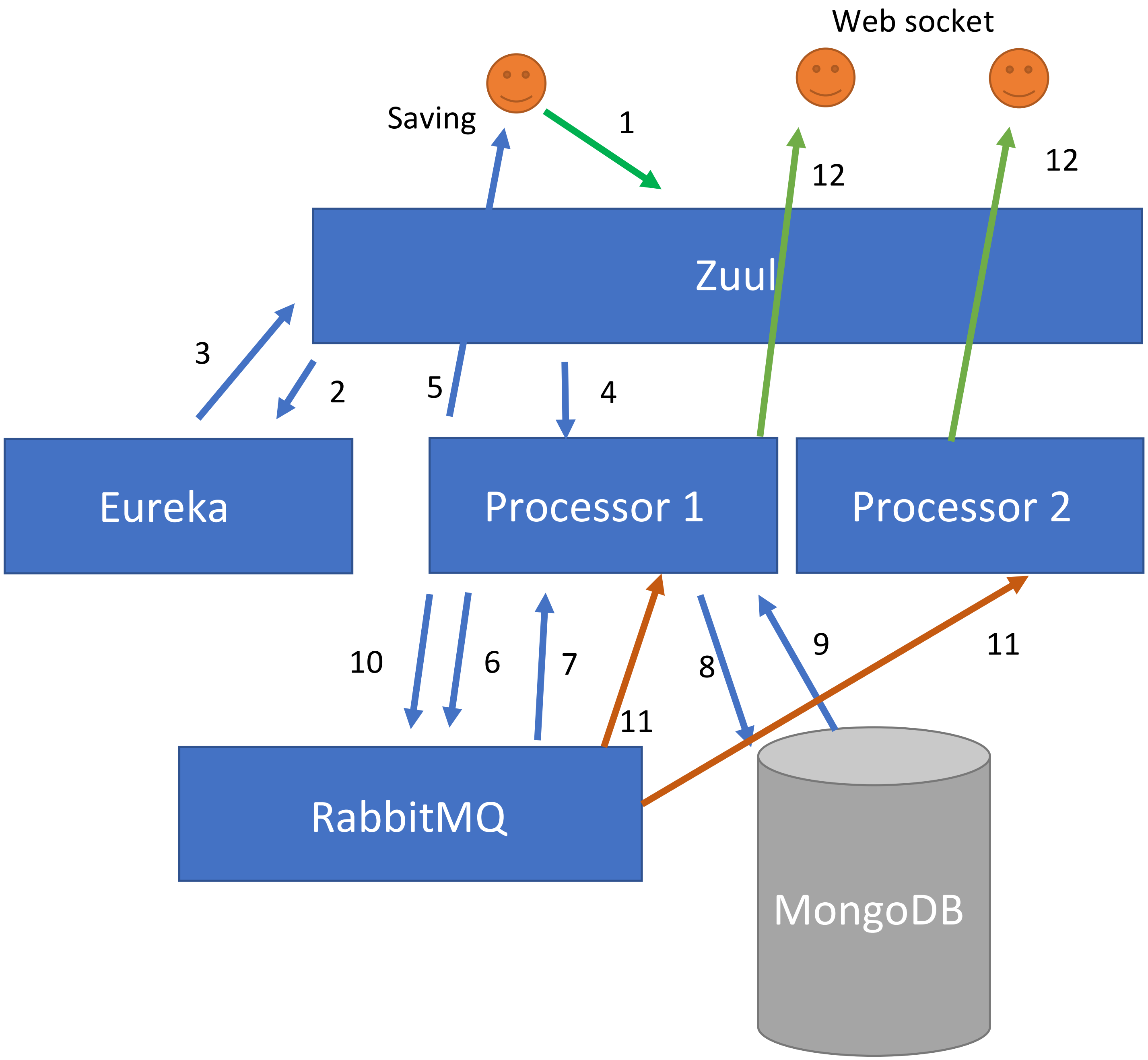
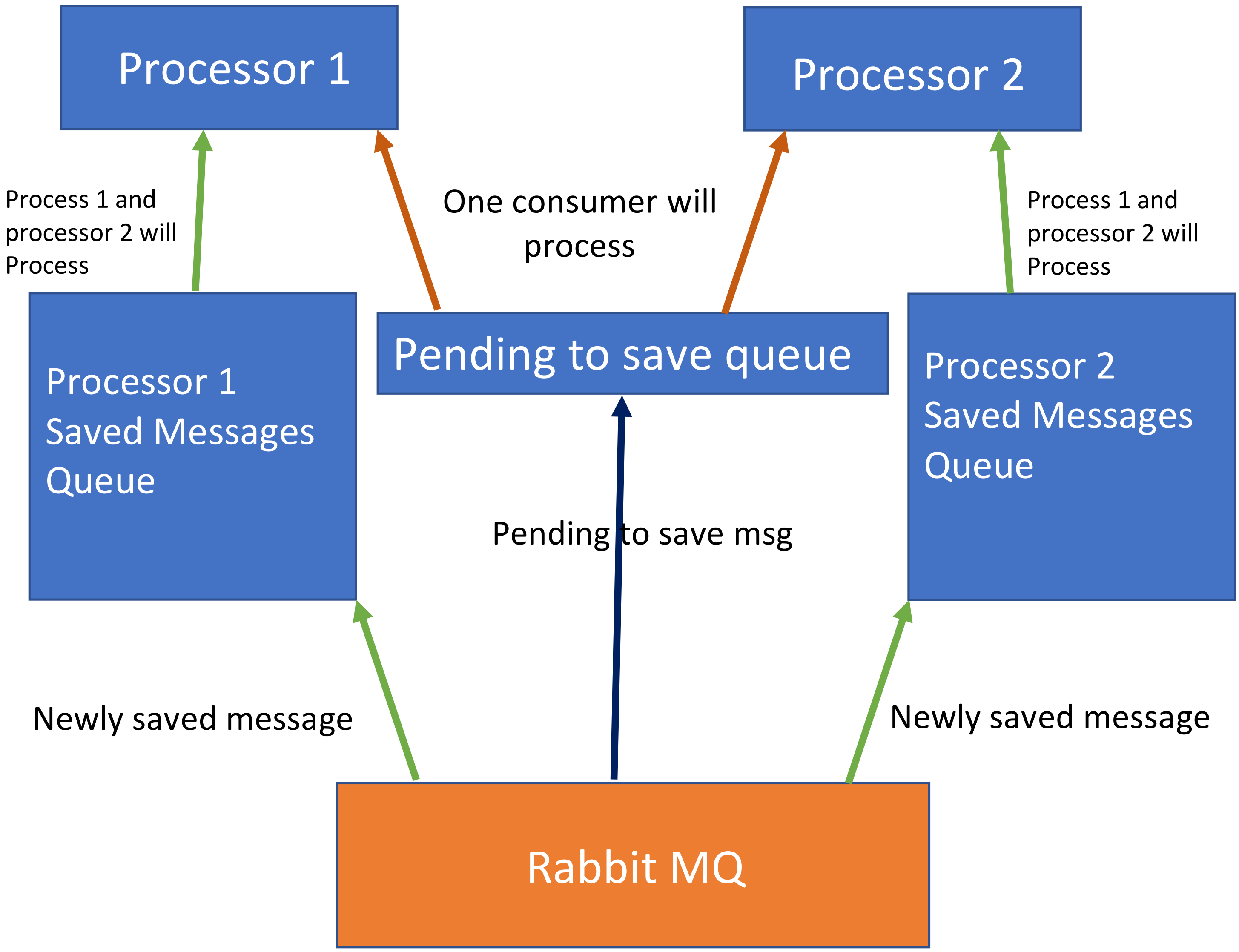
Processor Architecture



Steps:

1) User saves new element through the given Zuul URL
2) Zuul check on Eureka to see all the available instances
3) Eureka returns the URL of the instance
4) Zuul process it though a determinate Instance of processor
5) Processor receives the message and enqueues, returning a proper status to the user
6) (By Request) Processor enqueues to a persistence exchange of rabbit (Topic Type)
7) At this point RabbitMQ has a single exchange for pending to save messages that are binding to a single queue, the queue (on this example) has two processor client: Processor 1 and Processor 2, one of them will take the message and process it, in this example will be Processor 1
8) Processor 1 gets the message (if any extra logic to apply should be here) and then persist it at the Database (MongoDB)
9) DBMS returns OK to Processor 1
10) Processor 1 enqueues again the freshly saved messages (this is to show real time information on the web sockets doesn’t matter are multiple instances involved)
11) Rabbit receives the message on the same exchange (Check below to see RabbitMQ queue diagram), in this case, **we have one queue created by each instance of processor,** this will allow us to send the exact same message to both queues.
12) Processor sends new message to the channel where all web sockets are connected, sending the new saved message to all connected users

## Rabbit MQ


Processor 1
Processor 2
Process 1 and processor 2 will Process
One consumer will process
Process 1 and processor 2 will Process
Processor 1 Saved Messages Queue
Pending to save queue
Processor 2 Saved Messages Queue
Pending to save msg
Newly saved message
Newly saved message
Rabbit MQ