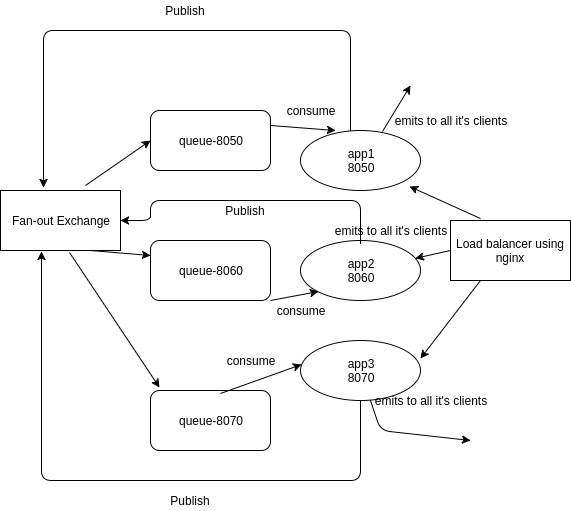
Socketio-game

1- I have used Node.js with express & socket.io for creating the game.

2- To make the game scalable, we have the following approach:

1. Use MessageBroker as Redis Pub/Sub
2. Use MessageBroker as AMQP

I will choose the second option to implement the scalable game.



Scalable Solution:

The current gihub solution is just a normal implementation of the game and it's not much scalable.

To create the scalable solution, the following things needs to be done.

1. App can be run on multiple ports.
2. Load balance & failover using nginx or haproxy.
3. Setup AMQP Message Broker like Rabbitmq and use amqplib npm in node.js to connect to RabbitMQ.

RabbitMQ implementation in node.js :

https://github.com/shaktiskm/es6-RabbitMQ

1. Each app has it's own queue for listening and will publish the messages to FAN-OUT exchange so that every message should go to all the binded queues with fan-out exchange.
2. For each app, client's message processing should be done via queue.
3. Queue messages should have all the sufficient information to process the message.
4. As every message reaches to all app's queues using FAN-out exchange, the processed result should be broadcasted by all the apps to it's client.