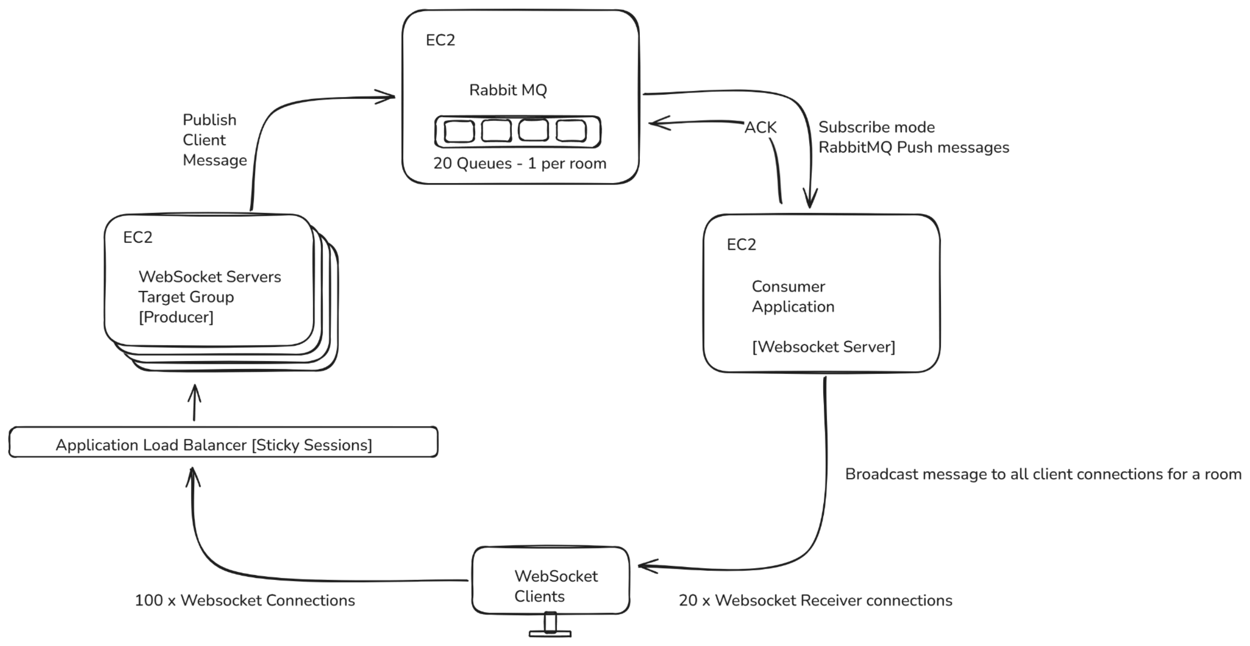
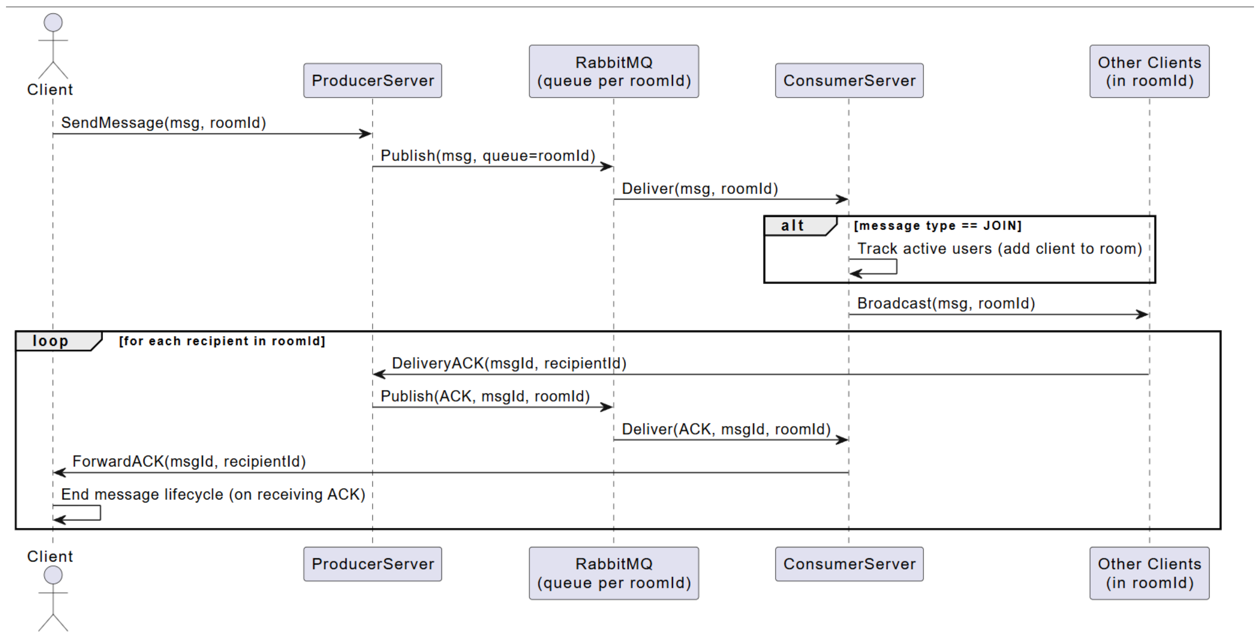
Krushna Sanjay Sharma
Source: https://github.com/krhash/chatflow/tree/main/chatflow-app/hw2

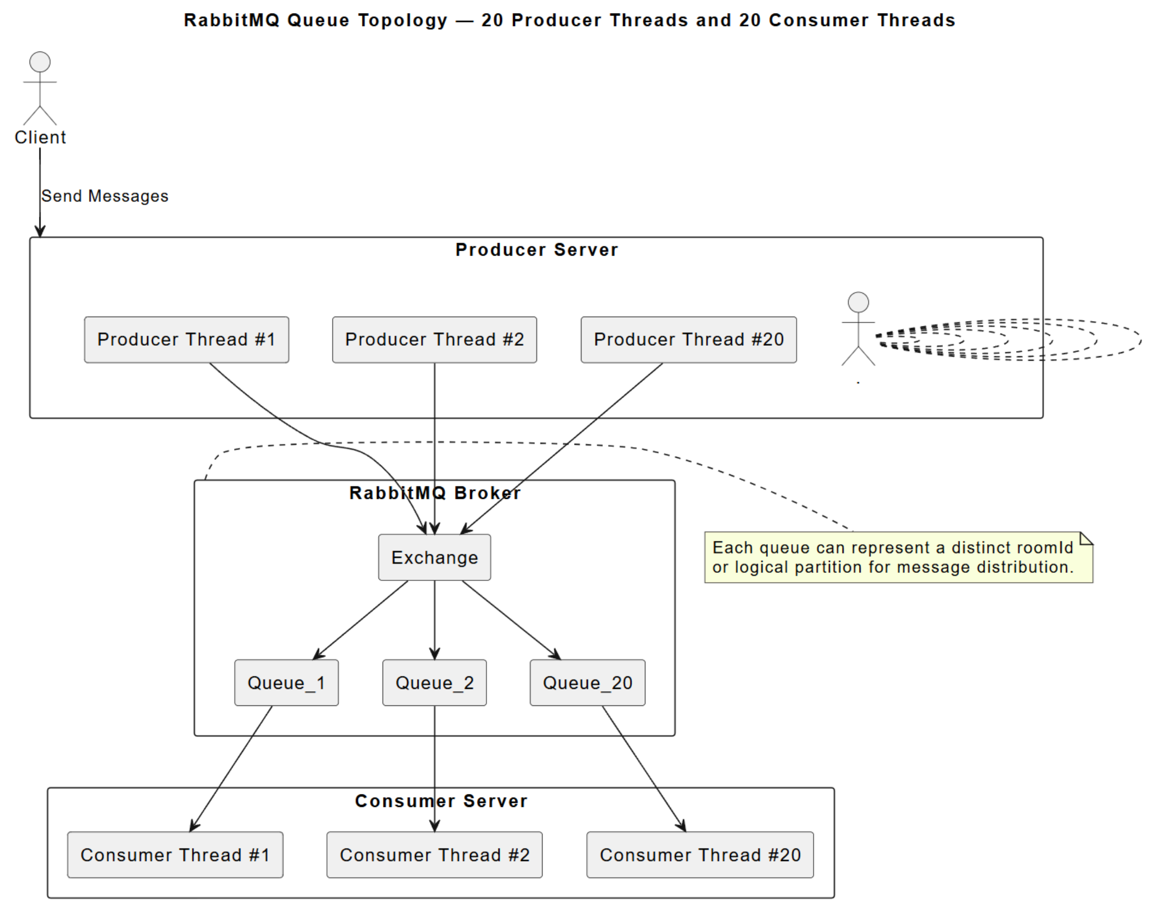
## Architecture Diagram:


EC2
Rabbit MQ
20 Queues - 1 per room
Publish Client Message
ACK
Subscribe mode RabbitMQ Push messages
EC2
WebSocket Servers Target Group [Producer]
EC2
Consumer Application
[Websocket Server]
Application Load Balancer [Sticky Sessions]
Broadcast message to all client connections for a room
WebSocket Clients
100 x Websocket Connections
20 x Websocket Receiver connections

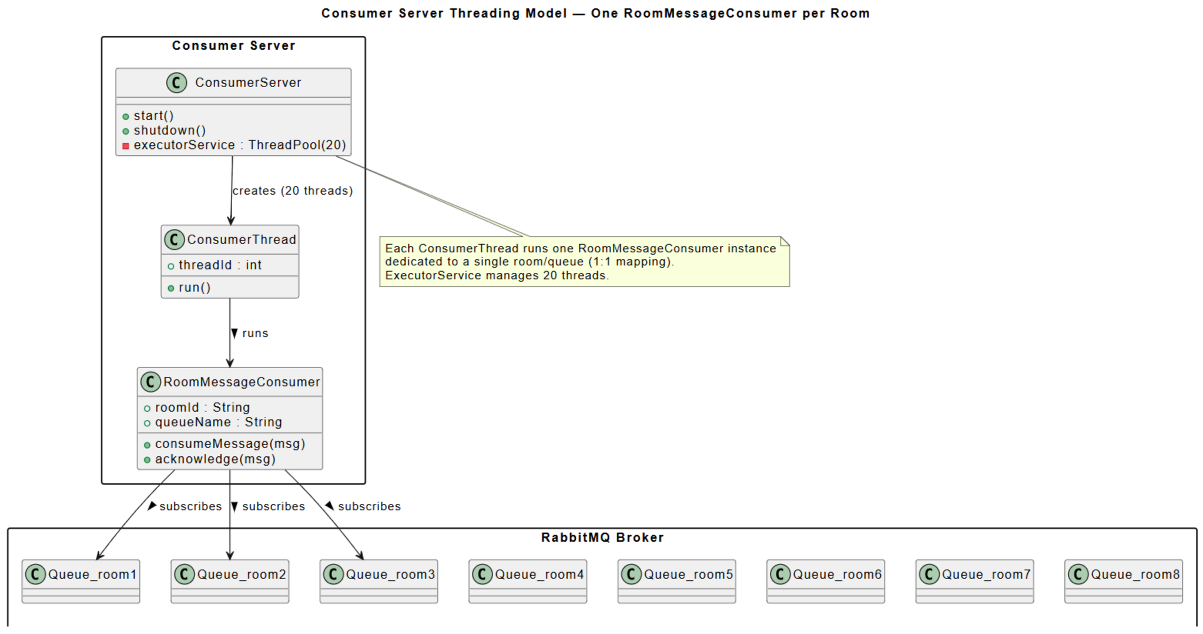
## Message Sequence Diagram:


Client
ProducerServer
RabbitMQ (queue per roomId)
ConsumerServer
Other Clients (in roomId)
SendMessage(msg, roomId)
Publish(msg, queue=roomId)
Deliver(msg, roomId)
alt
[message type == JOIN]
Track active users (add client to room)
Broadcast(msg, roomId)
loop
[for each recipient in roomId]
DeliveryACK(msgId, recipientId)
Publish(ACK, msgId, roomId)
Deliver(ACK, msgId, roomId)
ForwardACK(msgId, recipientId)
End message lifecycle (on receiving ACK)

## Queue Topology Design:

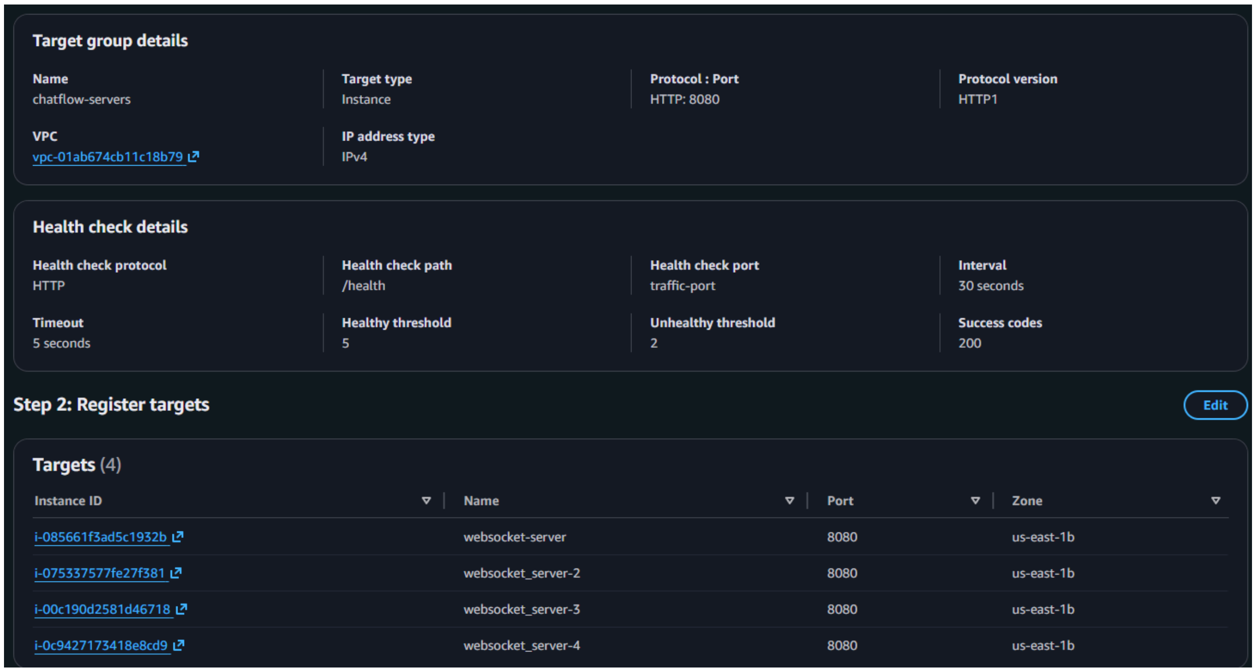


## Consumer Threading Model:



**Load Balancing Configuration:**

- Application load balancer with sticky sessions.
- 4 EC2 servers in the target gro



Target group details

| | | | |
|---|---|---|---|
| Name | Target type | Protocol : Port | Protocol version |
| chatflow-servers | Instance | HTTP: 8080 | HTTP1 |
| VPC | IP address type | | |
| vpc-01ab674cb11c18b79 | IPv4 | | |

Health check details

| | | | |
|---|---|---|---|
| Health check protocol | Health check path | Health check port | Interval |
| HTTP | /health | traffic-port | 30 seconds |
| Timeout | Healthy threshold | Unhealthy threshold | Success codes |
| 5 seconds | 5 | 2 | 200 |

Step 2: Register targets

Targets (4)

| Instance ID | Name | Port | Zone |
|---|---|---|---|
| i-085661f3ad5c1932b | websocket-server | 8080 | us-east-1b |
| i-075337577fe27f381 | websocket_server-2 | 8080 | us-east-1b |
| i-00c190d2581d46718 | websocket_server-3 | 8080 | us-east-1b |
| i-0c9427173418e8cd9 | websocket_server-4 | 8080 | us-east-1b |

**Failure prevention Strategies:**

1. Circuit breaker pattern on consumer-server to avoid cascading failures
2. WebSocket heartbeat from server to keep connections alive
3. Retry times out messages from client
4. Do not ACK messages that failed to broadcast, it stays on the message queue and get processed when another consumer thread wakes up.