SERVICE ORIENTED ARCHITECTURES

ACIT3855 – WINTER 2024

AGENDA

- Quick Review
- Quiz 5
- Asynchronous Messaging
- Lab 6
  - Setup Kafka and MySQL on VM
  - Integrate with your Receiver and Storage services

QUICK REVIEW

- What’s the difference between Synchronous and Asynchronous communication?
- What are the benefits of using messaging for communication between Microservices?
- What are some of the use cases of Apache Kafka?
- What are the names of the applications involved in a system using Apache Kafka? Where are the messages stored?
- What are some of the features of Apache Kafka?

QUIZ 5

- Quiz is on the Learning Hub
- Open book, do your own work
- You have <15 minutes to complete it

COURSE SCHEDULE

Week	Topics	Notes
1	• Services Based Architecture Overview • RESTful APIs Review	Lab 1
2	• Microservices Overview • Edge Service	Lab 2, Quiz 1
3	• Database Per Service • Storage Service (SQLite)	Lab 3, Quiz 2
4	• Logging, Debugging and Configuration • Storage Service (MySQL)	Lab 4, Quiz 3
5	• RESTful API Specification (OpenAPI) • Processing Service	Lab 5, Quiz 4
6	• Synchronous vs Asynchronous Communication • Message Broker Setup, Messaging and Event Sourcing	Lab 6, Quiz 5
7	• Deployment - Containerization of Services *Note: At home lab for Monday Set*	Lab 7, Quiz 6 (Sets A and B) , Assignment 1 Due
8	• Midterm Week	Midterm Review Quiz
9	• Dashboard UI and CORS	Lab 8, Quiz 6 (Set C), Quiz 7
10	• Spring Break	No Class
11	• Issues and Technical Debt	Lab 9, Quiz 8
12	• Deployment – Centralized Configuration and Logging	Lab 10, Quiz 9
13	• Deployment – Load Balancing and Scaling *Note: At home lab for Monday Set*	Lab 11, Quiz 10 (Sets A and B)
14	• Final Exam Preview	Quiz 10 (Set C), Assignment 2 Due
15	• Final Exam	

ASSIGNMENT 1

- Due Friday, Feb. 23rd at midnight
- Don't forget!
- You need to pick one of the three options of a real company that successful used microservices and a fictitious company in the same industry that failed using microservices

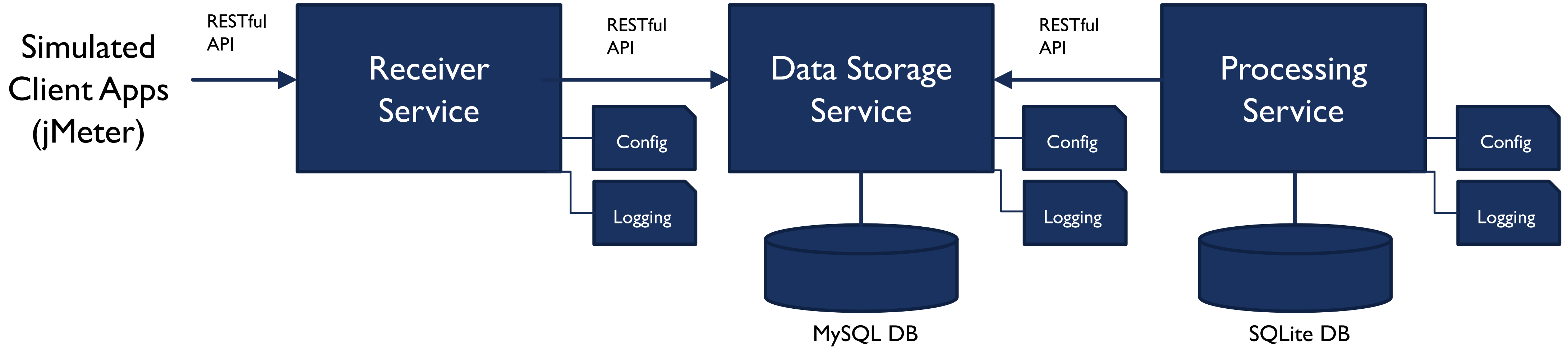
OUR SAMPLE APPLICATION

Our sample application will have three initial services:

- Receiver Service (Lab 2)
- Storage Service (Lab 3)
- Processing Service (Lab 5)

We will upgrade our services to enable scalability and availability:

- Database Service – MySQL (Lab 6A)
- Event Based Messaging – Kafka (Lab 6A/B)



SYNCHRONOUS VS ASYNCHRONOUS COMMUNICATION

Synchronous Communication – When a client makes a request and has to wait until the request fulfilled before receiving the response. The client thread generally is blocked until the server fulfills the request. The server thread has to complete all the activities required of the request before returning the response.

An HTTP call to a RESTful service is considered synchronous communication.

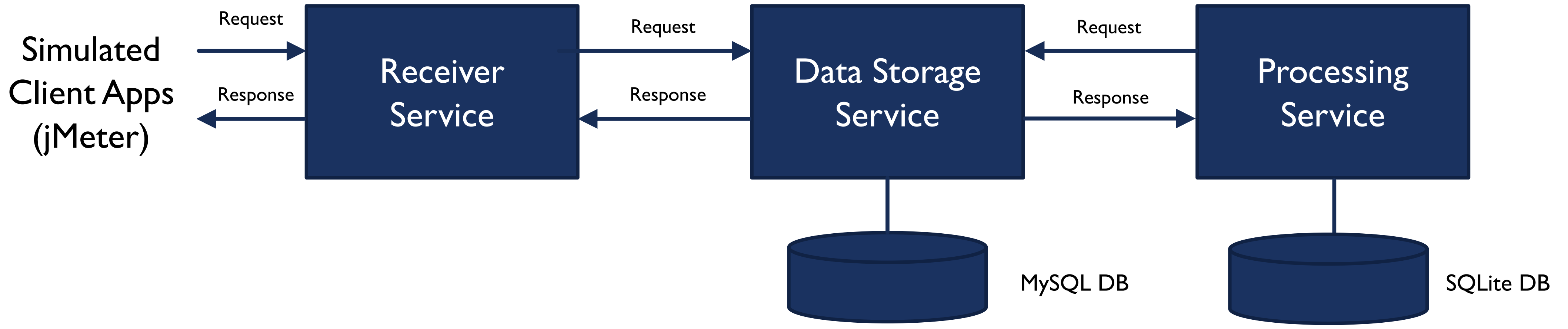
Asynchronous Communication – When a client makes a request and DOES NOT have to wait until the request completely fulfilled before receiving the response.

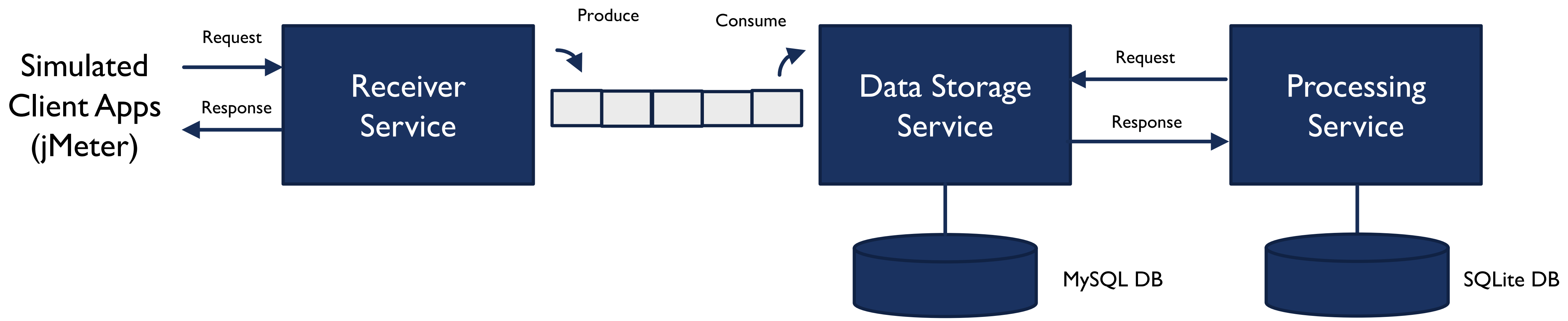
BENEFITS OF ASYNCHRONOUS MESSAGING (FROM THE READING)

- **Simple, scalable connectivity**
- **Simple, high availability**
- **Simple producer/consumer scalability**
- **The enablement of publish/subscribe, message filtering, routing and fanout**
- **Message rate and consumer availability decoupling**

Source: https://solace.com/blog/messaging-between-microservices/

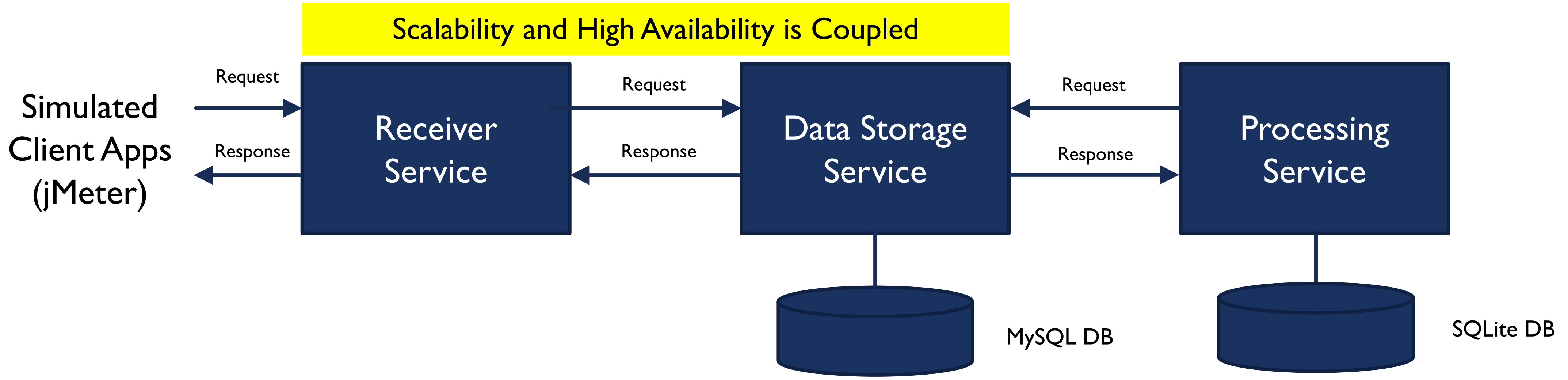
SYNCHRONOUS VS ASYNCHRONOUS COMMUNICATION

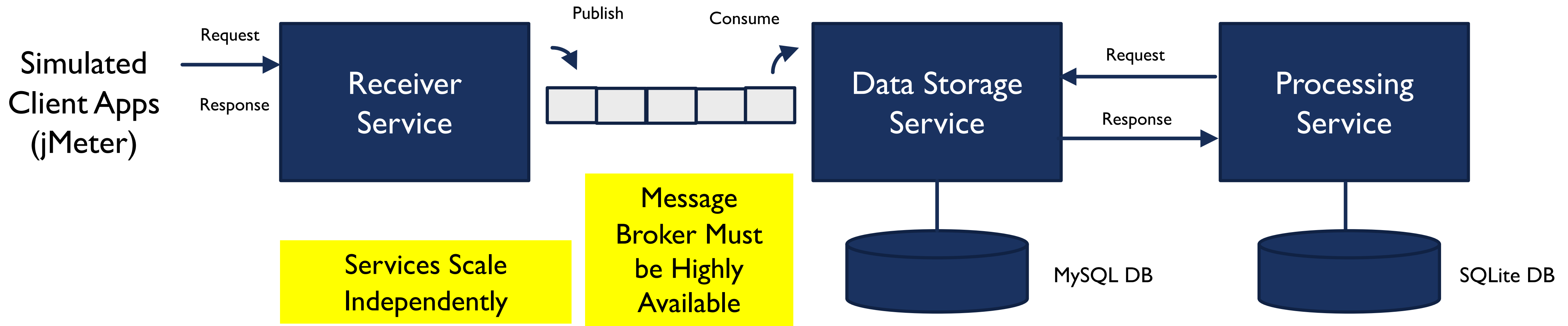




SYNCHRONOUS VS ASYNCHRONOUS COMMUNICATION

AVAILABILITY AND SCALABILITY





ASYNCHRONOUS COMMUNICATION - MESSAGE BROKERS

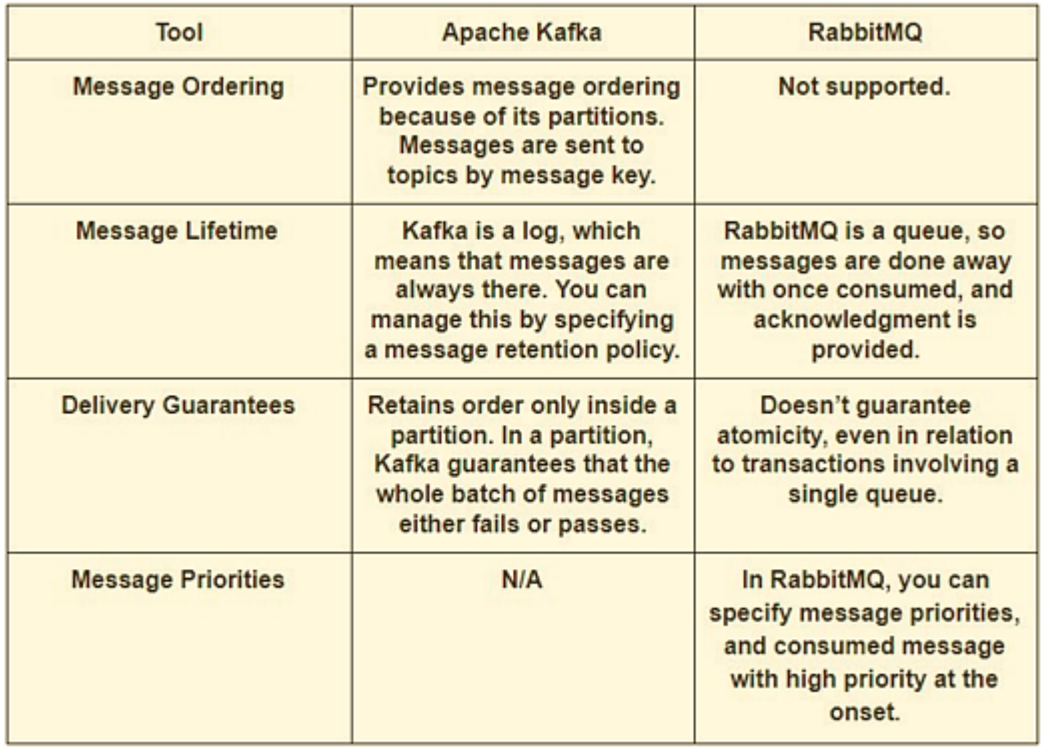
- A **Message Broker** is a software application that allows messages to be routed from one or more senders to one or more receivers. May also include validation and transformation of the messages.
- The senders are typically called Producers (or Publishers).
- The receivers are typically called Consumers (or Subscribers)
- A Message Broker may support one or more **Message Queues**. A Message Queue is a logical construct that transports specific types or groups of messages from the Producers to the Consumers.
- Producers publish messages to a Message Queue. Consumers subscribe to a Message Queue to receive messages.
- Examples:

APACHE kafka®

RabbitMQ™

ACTIVEMQ

APACHE KAFKA VS RABBITMQ



Tool	Apache Kafka	RabbitMQ
Message Ordering	Provides message ordering because of its partitions. Messages are sent to topics by message key.	Not supported.
Message Lifetime	Kafka is a log, which means that messages are always there. You can manage this by specifying a message retention policy.	RabbitMQ is a queue, so messages are done away with once consumed, and acknowledgment is provided.
Delivery Guarantees	Retains order only inside a partition. In a partition, Kafka guarantees that the whole batch of messages either fails or passes.	Doesn't guarantee atomicity, even in relation to transactions involving a single queue.
Message Priorities	N/A	In RabbitMQ, you can specify message priorities, and consumed message with high priority at the onset.

Source: https://www.upsolver.com/blog/kafka-versus-rabbitmq-architecture-performance-use-case

Apache Kafka Use Cases

- Streams with complex routing, throughput of 100K/sec events or more, with "at least once" partitioned ordering
- Applications requiring a stream history, delivered in "at least once" partitioned ordering. Clients can see a 'replay' of the event stream.
- Event sourcing, modeling changes to a system as a sequence of events.
- Stream processing data in multi-stage pipelines. The pipelines generate graphs of real-time data flows.
- Note: A Topic in Kafka is equivalent to a Message Queue. Topics can be further subdivided into Partitions.

RabbitMQ Use Cases

- Applications that need to support legacy protocols, such as STOMP, MQTT, AMQP, 0-9-1.
- Granular control over consistency/set of guarantees on a per-message basis
- Complex routing to consumers
- Applications that need a variety of publish/subscribe, point-to-point request/reply messaging capabilities.

MESSAGING

Microservices Principle:

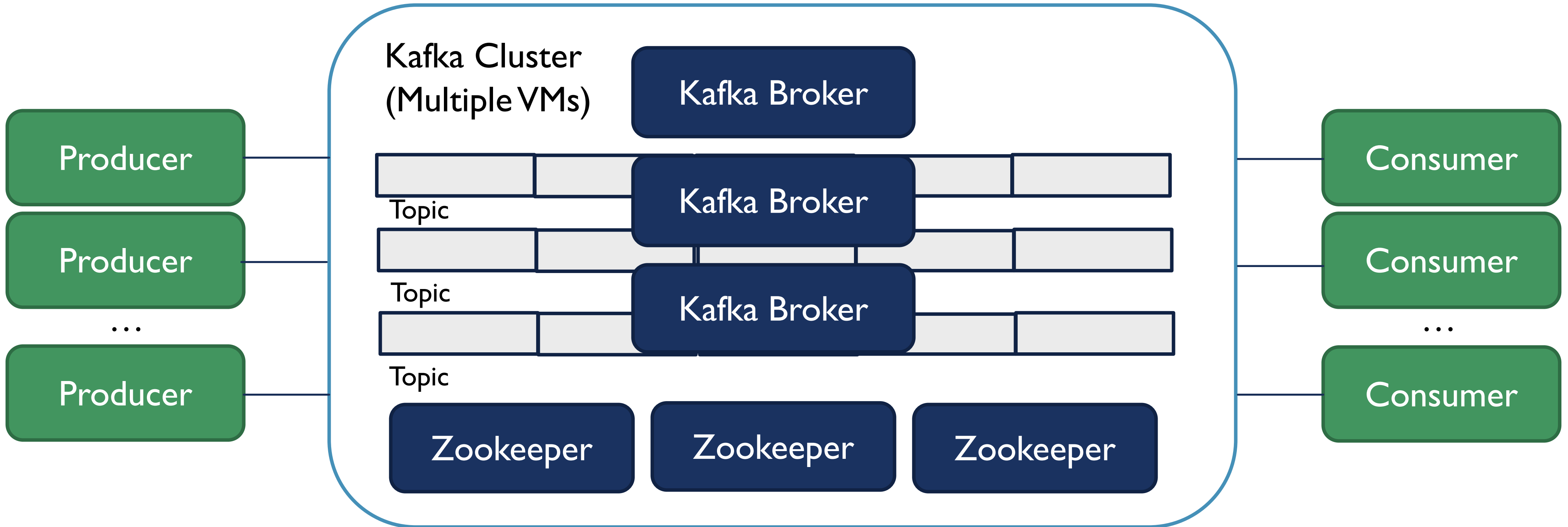
- Dumb Pipes and Smart Endpoints (more toward Kafka)
  - Pipe (i.e., Message Broker) is merely a highly-available transport mechanism (minimal logic)
  - Endpoints (i.e., Services) have the logic on how or if to process the messages

Traditional SOA: Smart Pipes and Dumb Endpoints (more toward RabbitMQ)

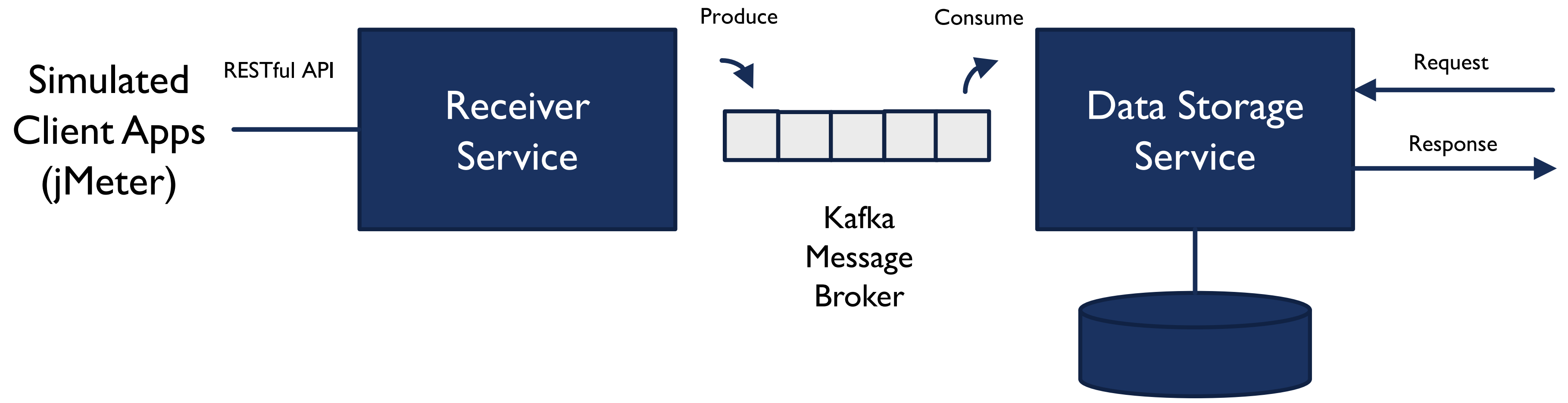
- Pipe (i.e., Message Broker) has complex logic – i.e., routing, retry
- Endpoints (i.e., Services) just process the messages they are given

KAFKA DEPLOYMENT

- Kafka is typically deployed in a Cluster
- There are multiple Kafka Brokers to allow for redundancy and high-availability (minimum 3 brokers for HA)
- There are multiple Zookeeper configuration stores that manage the state and configuration of Kafka (minimum of 3 Zookeepers for HA)



DATA STORAGE SERVICE - COMPLEXITY



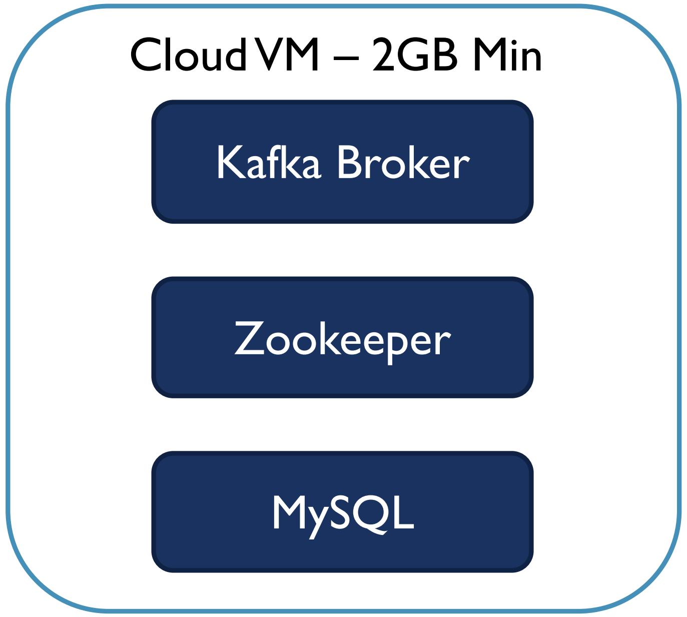
When we integrate the Data Storage Service with the Message Broker that means it has two key tasks:

- Consuming Messages
- Handling Requests on the RESTful API

Why might this be a challenge? How might we solve it?

TODAY'S DEPLOYMENT

- You will be deploying one Zookeeper, one Kafka Broker and a MySQL server to a Cloud VM using Docker and Docker Compose
- You'll integrate the MySQL server with your Storage Service
- You'll integrate the Kafka broker with your Receiver and Storage Services
- You'll also create a new service (Audit Log) that uses Kafka as a data store next week



TODAY'S LAB 6 – TWO PARTS

Lab 6A – Due by next class as screenshots only

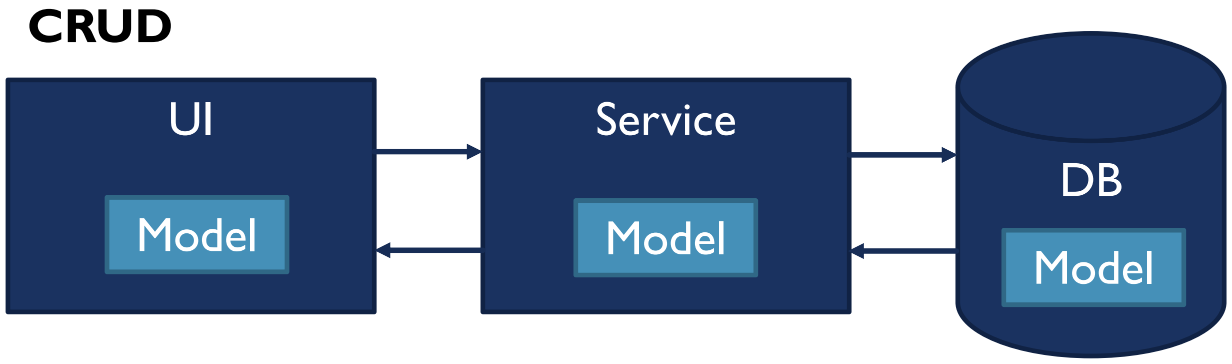
- Setup a VM on a cloud provider (Azure, AWS) with a minimum of 2GB of RAM (to support Kafka)
- Setup Zookeeper, Kafka and MySQL on that VM using Docker Compose

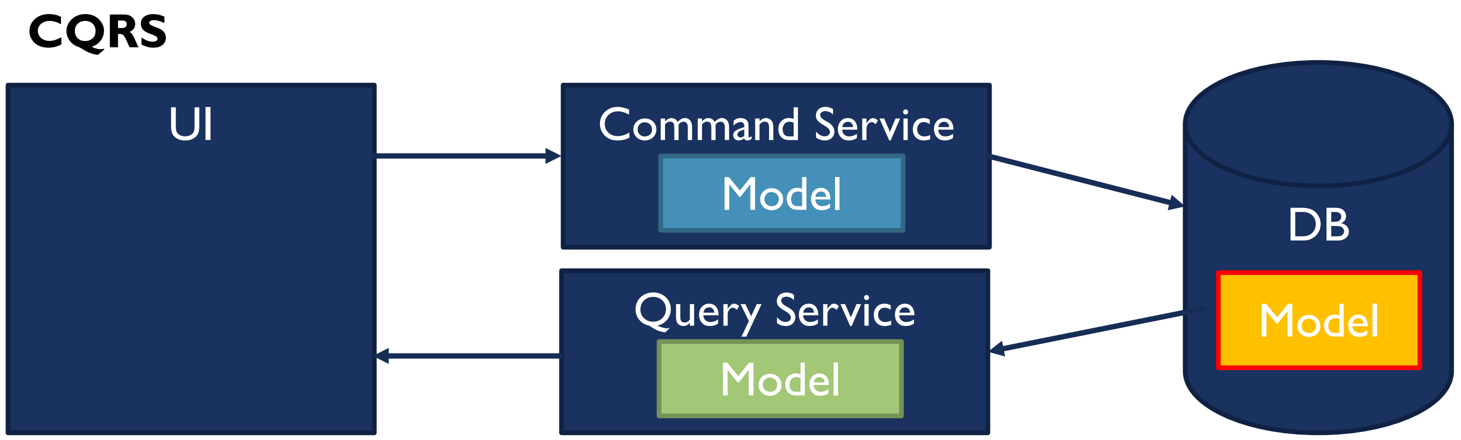
Lab 6B – Due for demo next class

- Integrate the Kafka broker with your Receiver and Storage Services
- Create a new service (Audit Log) that uses Kafka as a data store (Event Sourcing)

You will need the VM for subsequent labs, so don't delete it.

EVENT SOURCING

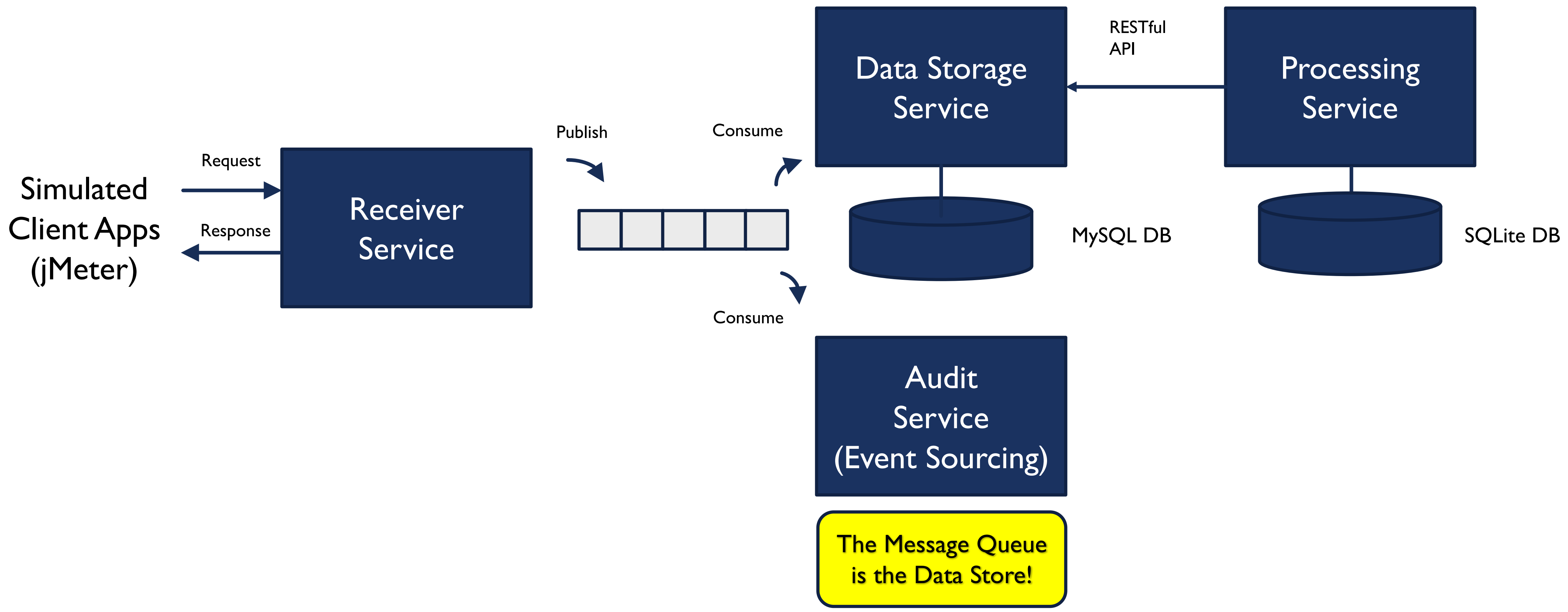




CRUD: DB could be an Event Store. Storing all events rather than replacing them. This allows for an audit trail and/or temporal queries (i.e., time based)

CQRS: Query Model is a Materialized View – so it is built from the DB model which could include historic events using ***Event Sourcing***.

LAB 6B


Simulated Client Apps (jMeter)
Request
Response
Receiver Service
Publish
Consume
Consume
Data Storage Service
RESTful API
Processing Service
MySQL DB
SQLite DB
Audit Service (Event Sourcing)
The Message Queue is the Data Store!