Lab 2 Assignment: Enhancing the Airbnb Prototype with Docker, Kubernetes, Kafka, AWS, and Redux

Due Date: November 24, 2025
Points: 40

This lab builds on your Lab 1 Airbnb prototype. You will enhance the application by containerizing it with Docker, orchestrating it using Kubernetes, adding Kafka for asynchronous message handling, and deploying it on AWS. Additionally, you will integrate Redux into the frontend to manage the state of user sessions, property data, and bookings.

Lab Overview

In this lab, you will extend your Lab 1 Airbnb project by deploying it using Docker, Kubernetes, Kafka, and AWS. You'll also integrate Redux into the frontend to manage the state of user sessions, property data, and bookings.

Part 1: Docker & Kubernetes Setup (15 points)

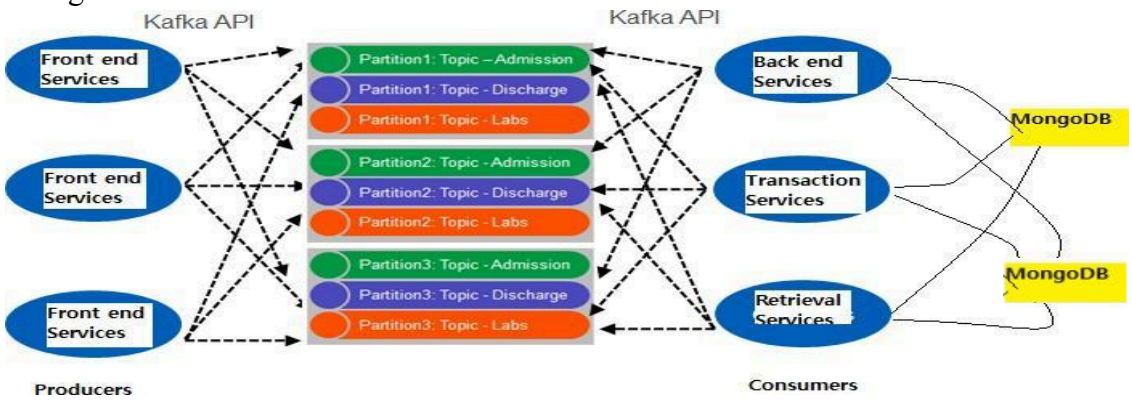
1. Dockerize your Lab 1 services: Containerize the Traveler service, Owner service, Property service, Booking service, Agentic AI service
2. Kubernetes Orchestration: Deploy the Dockerized services to a Kubernetes cluster. Ensure services can communicate with each other and scale properly.

Part 2: Kafka for Asynchronous Messaging (10 points)

1. Kafka Setup: Add Kafka to your Kubernetes setup to handle distributed messages.
2. Kafka Integration with Booking Flow: Implement Kafka to manage booking processing asynchronously, publishing events for booking creation and consumption by owner service for status updates.The flow:
    - Traveler creates a booking request → publish event to Kafka.
    - Owner service consumes the event to Accept/Cancel bookings.
    - Status updates are published back to Traveler service.

Separate Node into two parts connected via message queues.

Design “backend services” as consumer and “frontend services” as producer as shown below diagram.



Part 3: MongoDB (5 points)

Use MongoDB as your database. Sessions should be stored in MongoDB. Passwords

need to be encrypted.

Part 4: Redux Integration for State Management (5 points)

Redux Setup: Integrate Redux into your existing React frontend to manage state of:

- User Authentication: Store Traveler/Owner sessions (JWT tokens) in Redux.
- Property Data: Use Redux to fetch and store property information, such as property lists and details.
- Booking Data: Manage the traveler’s booking state in Redux, including favorites, booking status and updates.

Redux Store: Create a Redux store and define appropriate actions, reducers, and selectors to manage the application’s state across different components.

Example Redux Flow:

- Authentication: Dispatch login/signup actions to store the user's auth token in Redux
    - Traveler logs in → Redux stores JWT.
- Property Search: Use Redux to manage fetching and displaying property data.

    - Traveler searches for properties → Redux stores results.
- Order Process: Manage the state of items in the cart and track the order's progress through Redux
    - Traveler creates booking → Redux updates booking state.

Part 5: JMeter Performance Testing (5 points)

- Use Apache JMeter to test critical APIs in your application such as user authentication, property data fetching, and booking processing.
- Simulate concurrent Travelers making bookings and Owners responding. Measure response times, throughput, and error rates.
- Test the server for 100, 200, 300, 400 and 500 concurrent users. Draw the graph with the average time, your analysis of the graph on why, why not and how in the report.
- Submit JMeter test plan (.jmx file) and a summary of test results. Include screenshots of JMeter results and an analysis of performance bottlenecks.

Submission Guidelines

- Update your existing **Lab 1 GitHub repository** with:
    - Dockerfiles and Kubernetes configs for all services (Traveler, Owner, Property, Booking, Agentic AI).
    - Kafka integration code.
    - Redux frontend implementation.
    - JMeter test plans and results.
- Include:
    - Screenshots of services running on AWS.
    - Kafka event flow examples (Traveler → Booking → Owner).
    - Redux DevTools screenshots showing state changes.
    - JMeter results with performance analysis.

Report

Your report should cover:

- Describe how Docker, Kubernetes, Kafka, and AWS were integrated into your Lab 1 project.
- Explain how you integrated Redux into the frontend and how it improved state management.
- Provide screenshots of your services running on AWS and Kafka message flows
- Document how Redux is managing authentication, restaurant data, and order state