CMPE273: Enterprise Distributed Systems

Lab 2 Distributed Service using Kafka and MongoDB

Due: April 7th, 2019 11:59 PM

This lab covers designing and implementing distributed service-oriented application using Kafka and MongoDB deployed to AWS. This lab is graded based on 30 points and is an individual effort (no teamwork allowed)

Prerequisites:

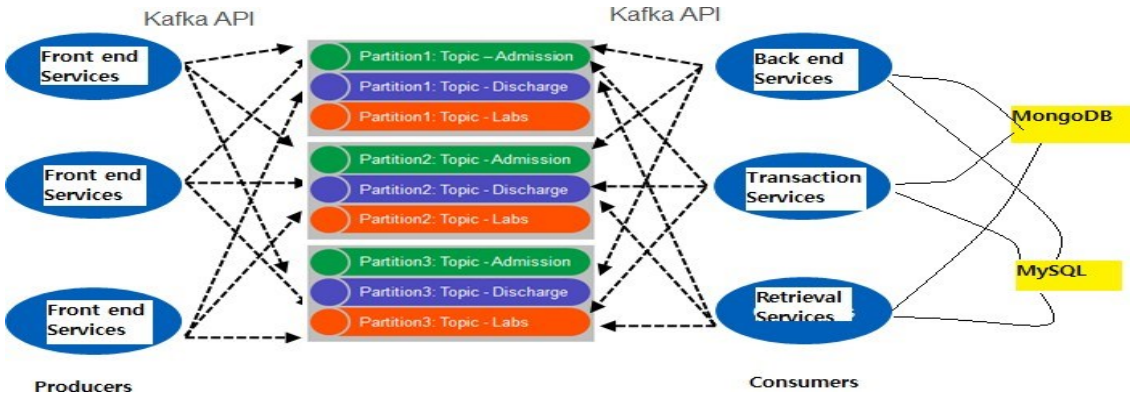
- You should be able to run Kafka sample example.
- You should have prior knowledge of JavaScript, default Sessions.

Grading

- Submissions received at or before the class on the due date can receive maximum
- Late assignments will be accepted, but will be subject to a penalty of -5 points per day late

The Assignment

- You will be developing a distributed client and server.
- Separate Node into two parts deployed in AWS connected via message queues. Design, implement and deploy "backend services" as consumer and "frontend services" as producer in the cloud as shown below diagram.



- Implement **connection pooling** as discussed in the lecture.
- Use MongoDB as the database.
- Passwords need to be encrypted (use **passportJS + JWT**).
- Use of Redux is mandatory for state management.
- Client and Server should communicate via **Kafka** Streams.
- Use of cloud services (AWS or Heroku) is mandatory.
- On, or before the due date, you have to turn in the following:
    - Code listing of client and server
    - Document with distributed cloud architecture of the Kafka interaction in your client/server application, system design description and screenshots

Canvas Application

Server - demonstrate RESTful Services (12 pts)

The node.js based server you need to develop is the "Prototype of Canvas application". Everyone should refer to Canvas and see how it functions.

This server should perform the following tasks:

1) Basic Users (Student & faculty) functionalities:
   a) Sign up new user (Name, Email and password)
   b) Sign in existing user
   c) Sign out.
   d) Profile (Profile Image, Name, Email, Phone Number, About Me, City, Country, Company, School, Hometown, Languages, Gender)
   e) Users can update Profile anytime.

   To use the system, a user must login first to the system. Password must be encrypted. Use PassportJS with JWT for session management. Use of Redux is mandatory.

   b) Faculty
      1. Only the Faculty can create course with fields CourseId, CourseName, Course Dept, description, CourseRoom, Waitlist Capacity, CourseTeam.
      2. Faculty should be able to give permission codes for waitlisted students.

   c) Home

      1. Student should be able to view all the courses he/she has registered.
      2. Faculty should be able to view all the courses created by them.

   d) Course Details:
      1. Student Details: Grades, Submit/view assignments, quiz, view announcements, people registered for the course, download lecture notes and files.
      2. Faculty details: Create assignments, quizzes, download submissions from students, make announcements, students registered for a course, grade assignments for students, upload lecture notes and files, remove student from a course.

   e) **Dashboard (New)**
      1. **The student dashboard with course cards should be made organizable.**
         a. **User should be able to move courses to top, bottom and right.**

   f) **Messaging feature (New)**
      - Messaging feature should be added to the Canvas application.
      - Students can message other students.
      - Students can message Faculty.
      - Show the messages under Inbox Tab.

f) **Pagination (new)**
   i) **Pagination should be added to People search, course search and other suitable views.**

The Service should take care of exception that means validation is extremely important for this server. **Proper exception handling and prototype like actual Canvas application would attract good marks.**

Client - [*4 pts*]
A client must include all the functionalities implemented by the web services. Develop the Client using HTML5 and ReactJS-Redux. A Simple, attractive and Responsive client attracts good marks.

Note: Every field in an entire project must have validation. User's Name (Navigate to Profile) etc. must have hyperlinks.

Hosting - [*7 pts*]
- **You must host your Canvas Application lab to cloud.**
- **You can user either Heroku or Amazon Web Service to host your client and server.**
- **You can use MLab for your MongoDB database.**

Testing - [*7 pts*]
Testing of the server should be done using JMeter and Mocha. **Testing of the front end should be done using Jest Enzyme.**

a. Following tasks to be tested using JMeter: (4 Points): Test the server for 100, 200, 300, 400 and 500 concurrent users (a) without connection pooling (b) DB provided connection pooling and. Draw the graph with the average time, your analysis of the graph on why, why not and how in the report.

b. Following tasks to be tested using Mocha: (1 Point): Implement five randomly selected REST web service API calls using Mocha. Display the output in the report.

c. **Following tasks to be tested using Jest Enzyme**: **(2 Points) Implement three randomly selected React components using Jest Enzyme. Display the output in the report.**

Questions (4 pts)

1. Compare performance with Kafka services, MongoDB deployed to cloud and local. Explain in detail the reason for difference in performance.
2. If given an option to implement MySQL and MongoDB both in your application, specify which data of the applications will you store in MongoDB and MySQL respectively

Deliverables Required (Git Deliverables):

- Inside the git repository assigned to you, create a folder, Lab 2.
- Inside this folder create three sub-folders, one for Frontend-Code and one for Backend-Code and one for Kafka. Place all your source code in respective Folders.
- Do not submit binaries, .class files, or supporting libraries (e.g., junit.jar, javaee.jar) (including them would be **3 points** deduction).
- Include the Readme file to document the steps to run the application.
- **All the dependencies should be added into package.json file.**

Submission (Report Submission)

- **On-line submission**: shall include your report (smith_lab2_report.doc). Submissions shall be made via Canvas.
- **Project report**
    - Introduction: state your goals, purpose of system,
    - System Design: Describe your chosen system design
    - Results: Screen image captures of each client/server pair during and after running.
    - Performance: What was performance? Analyze results and explain why you are getting those results.
    - The answers to the questions.