System design architecture for hotel booking apps (Like Airbnb, OYO)

How do hotel booking applications like Airbnb, Booking.com and OYO work to provide such a smooth flow, from hotel listing to booking, to payments? And all without a single glitch! In this blog, you will get a detailed explanation for this.

As these are so huge that they have a high amount of user traffic that need to be processed. So to manage these we have to follow micro-service architecture. Which mean we have to divide the system into small chunks for each type of task.

Let’s understand the flow one by one. I have divided it into 4 parts:

- Hotel Management Service
- Customer Service (Search + Booking)
- View Booking Service

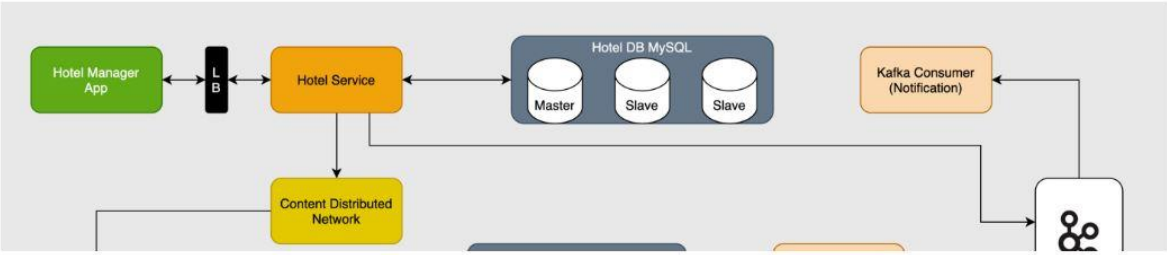
Hotel Management Service

This is the service that will be given to hotel managers/owners. In this managers can manage their hotel's related information. Here managers have a separate portal to access the data and update it.

Whenever an API is triggered from the hotel manager app the initial request is been sent to the load balancer, then the load balancer distributes the requests to the desired server to process. The hotel service cluster has multiple servers that have the container for hotel service-related API.

Now, this hotel service interacts with the Hotel DB cluster which follows the master-slave architecture to reduce the load in the database. Basically, in this approach, we create a replica of the master database which are called a slave database. Master DB is used for a write operation and slave DB is used for reading operation only. Whenever a write operation is performed on the master database it syncs the data to the slave database.

Whenever any data is updated in the database API sends the data to the CDN(Content Distributed Network) and to a Messaging Queue System(like Kafka, RabbitMQ) for further process. A CDN is a geographically distributed group of servers that work together to provide fast delivery of Internet content.



Hotel Management Service Architecture

Customer Service (Search + Booking)

This is the service that will be given to customers. In this customers can search and book a hotel. Here customers have a separate portal to access the data and process it.

The CDN app shows the content to customers like the nearby hotels, recommendations, offers etc.

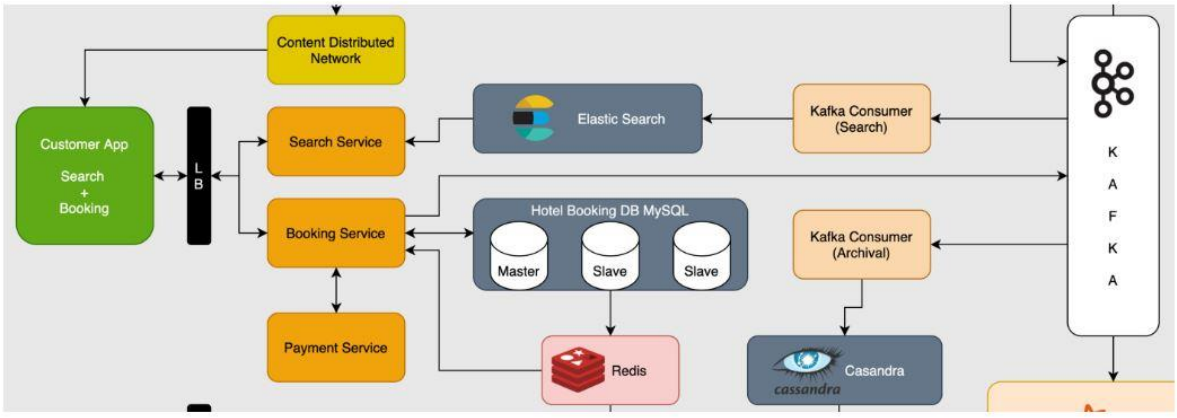
As we discussed in the previous section, hotels data is sent in messaging queue system to process it. Here we have a messaging queue consumer that takes the data from the queue and store the data in elastic search.

Customer app hit's the API then load balancer redirect and distribute the request to respective service to process the request. Here we have two services one for searching hotel and booking service to book the hotel and booking service also interact with the payment service that will be a third party service.

The search service has to get's the data from Elastic Search. Elasticsearch is a NoSQL Database that is best for its search engine functionality.

The booking service communicates with Redis and the booking database cluster. Redis is caching system, that's stores temporary data so that data need not fetched database and which could eventually reduce the load in the database also reduce the response time of API.

Any changes made in the database will be sent to the messaging queue. Then the consumer will take the data from the queue and put it to Casandra. For archival we are using Casandra because with time data size will increase in the database which will increase query time. So that's why we may need to delete old data from the database. And Casandra is a NoSQL database that is good at handling a high volume of data.

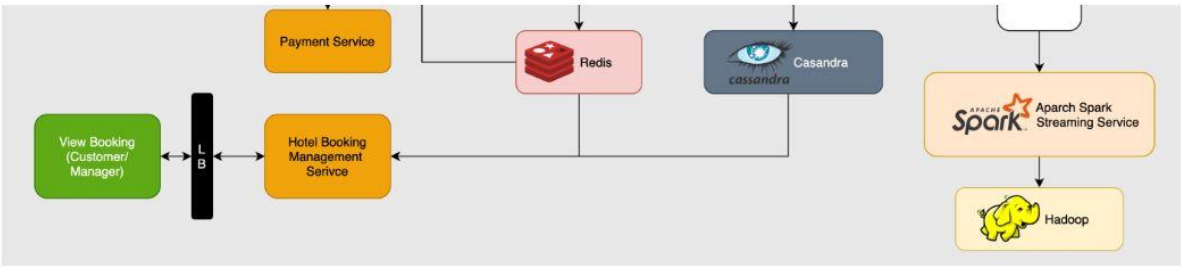


Customer Service Architecture

View Booking Service

Here all current and old booking details are shown to the user. This service is used by both managers and customers.

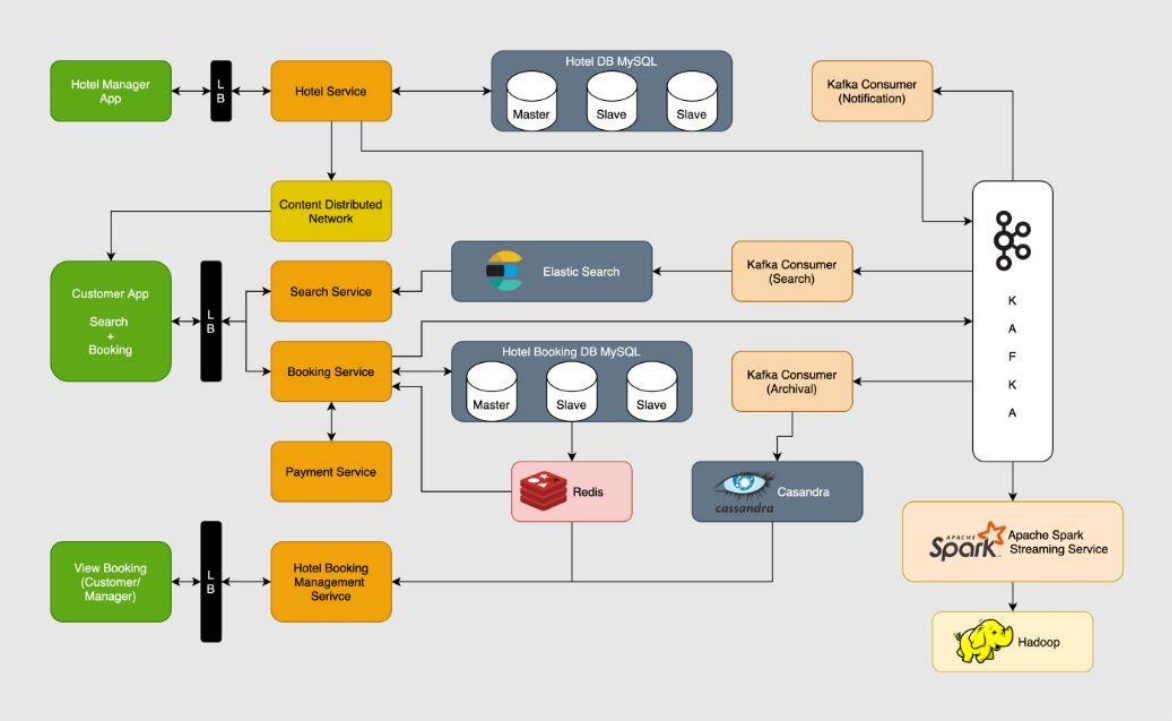
The Customer/Manager app sends the request to the load balancer and it distributes the request into booking management servers. Then the service request for data through Redis and Cassandra. through Redis, it requests the recent data as it is a caching server. Which could reduce the loading time on the app side.



View Booking Architecture

As you can see in the above design there is a Kafka consumer for notification, notification consumers send the notification. That could be to customer/manager, like whenever a customer books a hotel notification sends to the manager or if a new offers come it's notified to the customer.

Apache Streaming service takes the data from messaging queue and store it in Hadoop that could be used for BigData analysis for multiple purposes. Like business analysis, finding potential customers, audience categorisations etc.



Hotel Booking System Design