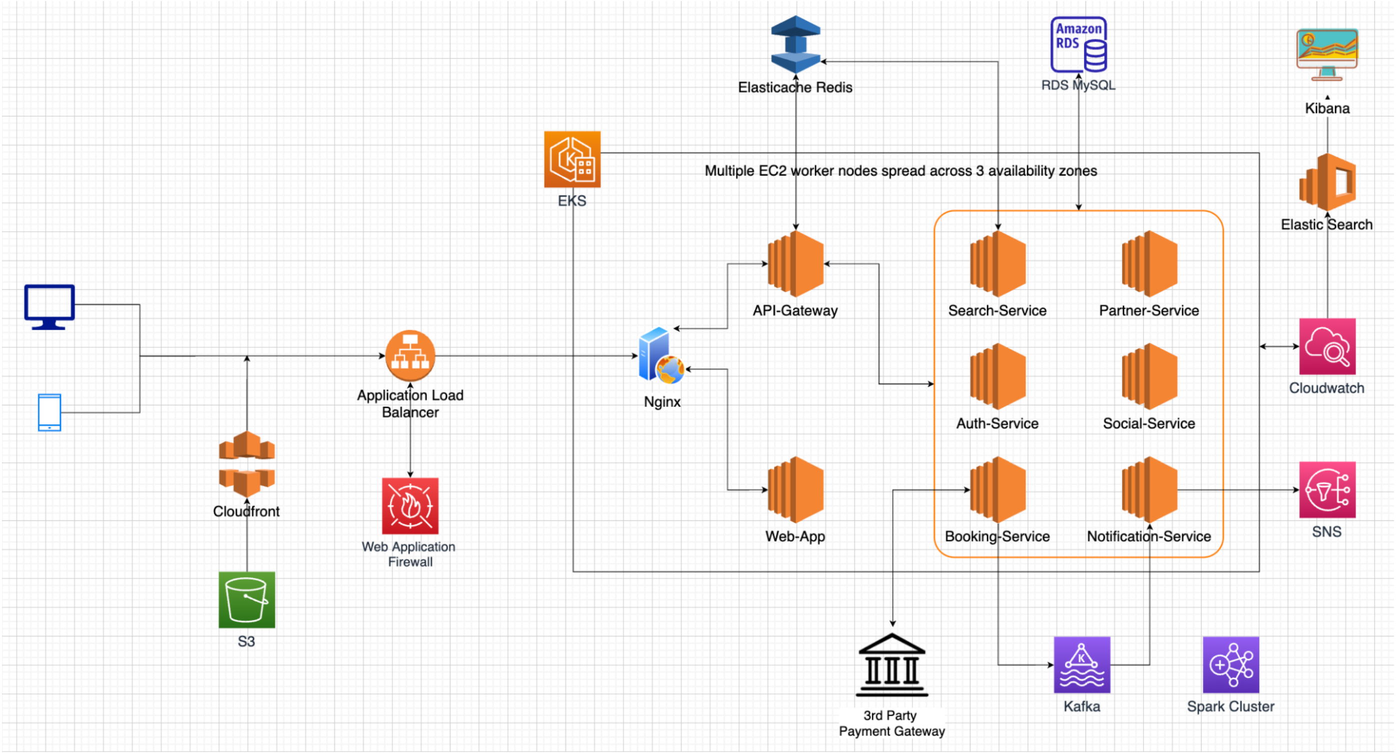
Architecture



- The above architecture is part of 1 AWS region spanning across different availability zones to ensure resiliency and availability
- Depending upon the traffic, the architecture can be extended to different region and load balanced via AWS Route53
- It addresses network traffic from desktop and mobile users
- EKS is a core managed service providing container orchestration for kubernetes. Benefits of EKS can majorly utilize w.r.t scalability. Alternative to EKS can be ECS on EC2 or Fargate. With EKS we can ensure architecture is cloud agnostic.
- Every service and web application will bundled into into docker image and pushed into image repository on AWS i.e. ECR

Core Components

Cloudfront

- It is CDN that will help to serve static contents on web application
- It provides low latency by caching static contents in edge locations
- Static contents will be stored in S3 bucket and distribution can be configured with origin as S3 bucket to serve static contents

Application Load Balancer

- ALB will act as a front gateway for application. Multiple targets can be configured via Rule; thus helping distribute traffic for commercial users and partners to different targets
- EKS cluster security group will be configured to listen request from ALB only making ALB entry gate for requests

Web Application Firewall

- WAF can be integrated as ALB preventing application from various attacks
- WAF can help control
  - XSS Protection
  - SQL Injection
  - IP address blocking
  - Provides CAPTCHA
  - Many more

Nginx

- It is web server helping distribute / distinguish traffic between Web-App and Microservices / API
- Alternative to Nginx is Apache web server but Nginx beats in terms of performance due to its nature of handling multiple connections concurrently

Elasticache

- It is a fully managed in-memory datastore making it a good candidate for caching API results
- It will be used by “Search-Service” microservice to cache results

- As an alternative, we can have a Redis pod running inside the kubernetes cluster. Maintenance and monitoring of the Redis will be overhead hence managed service is a better choice here.

MySQL

- It will be used to store all transactional and business data
- Advantage of managed service for MySQL is to
  - Scale
  - Read-Write replicas
  - Automated backups
- Alternative to RDS MySQL can be MySQL kubernetes pod running inside the EKS cluster. Volume for MySQL can be mounted to EBS volume for persistent storage

Kafka

- Based on booking confirmation, end user needs to be notified about the confirmation and other details w.r.t booking
- It is important to have a queue to ensure events / messages are delivered to end users. This is will be managed by consumers running inside kube cluster to ensure messages are delivered to SNS

Spark Cluster

- Spark cluster will help create reports by processing data in distributed manner and storing the processed data in MySQL
- These reports will provide better insight to partners e.g.
  - Revenue by location / Revenue by month
  - Repeat customers
  - Other products revenue (e.g. Meals)
- Kafka along with Spark streaming can help pre-process data based on upserts and storing data in cache

SNS

- This is managed service for communication with flavors of SMS, Email and Push notifications

Microservices

- As part of coding exercise delivered below are some of assumptions

  Out Of Scope

  - Authentication (Auth Service) & Authorisation (Part of API-Gateway)
  - Social micro-service that helps capture user reviews and feedbacks
  - Booking service and 3rd party payment gateway
  - Notification service
  - Config service - Helps manage microservice configuration

- Partner Service
  - This service will only cater to users part of B2B. Access will be controlled based on RBAC implementation by distinguishing B2C and B2B users
  - It will expose various endpoints helping B2B users perform
    - Manage theater screens & seats
    - CRUD operation for shows for available movies

- Search Service
  - This service explicitly address search queries from UI / Client
  - It is responsible to query datastore and have results cached in Elasticache
  - This microservice is built for concentrating only search features considering the majority of requests will be routed to this microservice.
  - Horizontal scaling of search service can be done independently depending on requests and CPU utilization by pod

- API Gateway
  - It is a JVM based routing offered by spring framework and is an important component of microservice architecture
  - It helps routing request to specific microservice depending on context path
  - Filters can be added to have authorization checks on protected endpoints / microservices
  - Optionally it can be coordinated with Redis for session management

Monitoring

- Application monitoring is done by integrating cloudwatch logs with elastic search datastore (OpenSearch). Kibana is used for visualization
- Cloudwatch alarms are configured to monitor cluster health. It provides insights on CPU Utilization, RAM Utilization etc.
- Network monitoring can be added by integrating Container Insights available part of cloudwatch
- Alarms can also be used for raising alerts in the event of multiple error in a given log group / pod

Project Planning

- Planning will start with task list required to achieve proposed architecture
- Assigning ballpark estimates to each tasks
- Considering resource / developer available with required skill set come up with gantt chart providing tentative timelines for project
- Compute implementation / development cost with derived efforts and timelines
- Compute operational cost of proposed architecture by using AWS calculator
- Both implementation cost and operational cost will help stakeholders take effective decision and plan budget accordingly