Booking Platform

Functions:

1. Touchpoint - Admin: [Authentication using Credentials]

1) Add movie
2) List movies - With Pagination - Order by UpdatedDate
3) Search movies - With Pagination - Order by UpdatedDate
4) Update movie
5) Delete movie
6) Add Partner
7) Update Partner - Update Details, Activate, Deactivate
8) Delete Partner - Soft Delete
9) Search Partners - With Pagination - Order by UpdatedDate
10) List Partners - With Pagination - Order by UpdatedDate
11) Add Partner User
12) Update Partner User - Update Details, Activate, Deactivate
13) Delete Partner User
14) Add Admin User
15) Update Admin User
16) Delete Admin User
17) Create offers
18) Update offers

2. Touchpoint - Partner: [Authentication using Credentials]

1) List movies - With Pagination - Order by UpdatedDate
2) Search movies - With Pagination - Order by UpdatedDate
3) Create Show
4) Update Show
5) Delete Show
6) Add Partner User
7) Update Partner User
8) Delete Partner User

3. Touchpoint - User: [Open to Internet, OAuth if user prefers to login]

1) List shows for today - Default call
2) Search shows for given dates
3) Get show details
4) Get offers
5) Book tickets - single or bulk
6) Make Payment
7) Get Booking Info
8) Cancel Ticket
9) View Booking History

4. Inbound/Outbound with Partner Systems

5. Vendor Integration - Payment Gateway, SMS Notification

MicroServices:

1) User
    - Database - RDBMS
    - Database - NoSQL for Partner facilities
    - Functions - Rest APIs with Spring Authorisation
    - Open APIs - Login
2) Movies
    - Database - RDBMS
    - Database - NoSQL for Movie Details
    - Functions - Rest APIs with Spring Authorisation
    - Open APIs - List Movies, Search Movies
3) Transaction
    - Database - RDBMS
    - Functions - Rest APIs with Spring Authorisation
    - Open APIs - List Shows, Search Shows, Get Show Details

Authentication & Authorisation:

1. Stateless Authentication & Authorisation using JSON Web Tokens
2. OAuth for EndUser Login
3. Authorization using Spring Roles

Configuration Management:

1. Spring cloud config

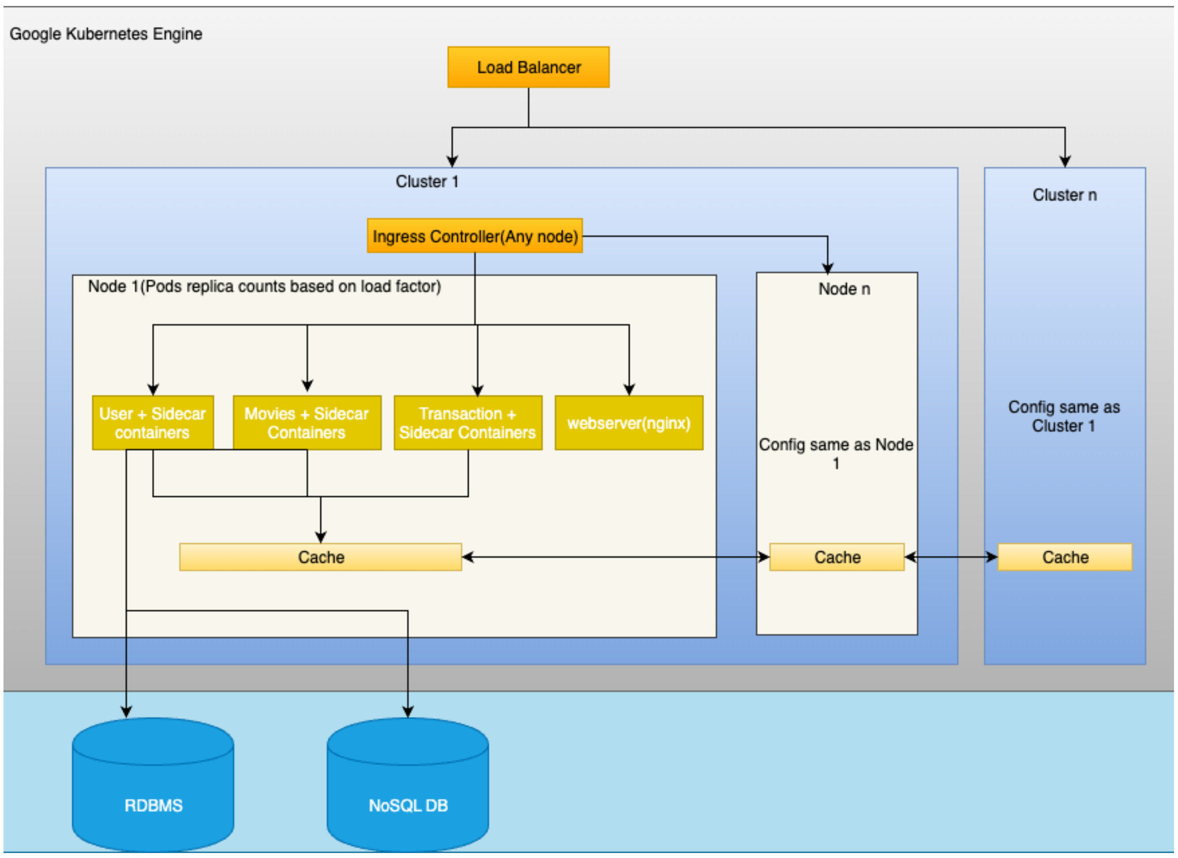
Logging:

1. Log4j2 logging - Publish logs to Splunk via Fluentd side car containers

Monitoring:

1. Grafana monitoring for pods
2. Dynatrace/Geneos monitoring

Platform:


Google Kubernetes Engine
Load Balancer
Cluster 1
Ingress Controller(Any node)
Node 1(Pods replica counts based on load factor)
User + Sidecar containers
Movies + Sidecar Containers
Transaction + Sidecar Containers
webserver(nginx)
Cache
Node n
Config same as Node 1
Cache
Cluster n
Config same as Cluster 1
Cache
RDBMS
NoSQL DB

Architecture

Data Model:

Timezone of Database & Servers - Asia/Kolkata

Movies:

Id	Name	Language	Updated Date
1	Kantara	Tamil	22/10/2022 10:00:00
2	Kantara	Kannada	22/10/2022 10:00:00
3	Kantara	Hindi	22/10/2022 10:00:00

Movie Details: (NoSQL):

```
{
Id:1
name: Kantara
actors: Rishab
synopsis: Test
image: /imagerepo/katanra.jpg
trailer: /videorepo/kantara.mp3
ratings: 4.5
},
{
Id:1
name: Kantara
actors: Rishab
synopsis: Test
image: /imagerepo/katanra_kannada.jpg
trailer: /videorepo/kantara_kannada.mp3
ratings: 4.5
}
```

Partners:

Id	Name	Address
1	Alankar	Alankar Address
2	PVR	Forum Mall

Partner Inventory:

Id	PartnerId	Hall	Capacity
1	1	1	50
2	1	2	60
3	1	3	70

Partner Inventory Seat Map:

```
{
PartnerId: 1,
InventoryId:1
rows:[
        A:{1,2,3,4,5,6,7,8},
        B:{1,2,3,4,5,6,7,8},
        C:{1,2,3,4,5,6,7,8},
        D:{1,2,3,4,5,6,7,8}
],
column_breaks:[4]
}
```

Shows:

Id	PartnerId	MovieId	Hall	Class	Show Time	Available Seats	Blocked Seats	TicketPrice
1	1	1	1	Balcony	21-10-2022 13:30	A1,A2,A3,A6, A7,A8,B1,B2, G1,G3,G4	A1,A2	150
2	1	1	2	Normal	21-10-2022 13:30	A1,A2,A3,A4, A5,A6,A7,A8, B1,B2,G1,G3, G4		150
3	1	2	3	Normal	21-10-2022 13:30	A1,A2,A3,A4, A5,A6,A7,A8, B1,B2,G1,G3, G4		150
4	2	1	1	Normal	21-10-2022 16:30	A1,A2,A3,A4, A5,A6,A7,A8, B1,B2,G1,G3, G4		200

Offers: (Query for partner/flat offers - Match the offer if all conditions of offer is satisfied)

```
{
Id:1
ticket:3
status:A,
updatedDate: 22/10/2022 10:00:00
},
{
Id:2
partnerId: 1
status:A,
updatedDate: 22/10/2022 10:00:00
},
{
Id:3
showdate: 22/10/2022
showtime: 13:30
status:A,
updatedDate: 22/10/2022 10:00:00
}
```

Bookings: (Clear draft bookings in few days)

Id	UserId	MobileNo	Email	Booking Number	Show	Seats	Offer Applied	Booking Dttm	Status
1		987463654	j@j.com	XYZ001	1	A1,A2	1	22/10/2022 10:00	Booked

Payments:

Id	BookingId	Amount	Status	VendorReferenceNumber	Mode	PaymentDttm
1	1	300	Paid	RAZOR123	UPI	22/10/2022 10:00

Blocking Seats Logic:

1) Get Availability - Get seats from Database for the selected show - Get blocked seats from Cache for the selected show - Filter blocked seats and send remaining seats

2) Book seats - Add seats to blocked seats in Cache if its free, If its already in blocked list, respond to user that seat is blocked - Checking and adding to cache should happen in synchronised block to maintain one truth

3) Once Booking is completed, seats will be removed from available seats in database, hence it can removed from blocked seats in cache

4) Cache key - PartnerId, HallId, MovieId - Blocked Seats - This expires in 10 mins

Performance Considerations:

1. Always load only required details from DB - Page, Scroller based loading
2. Low latency - First load required details, load other details later
3. No long-running complex queries, it should be split into multiple queries
4. Use Materialised views for history data
5. Use Indexes wherever required
6. Check whether DB connections are properly closed - Compare pool settings with DB configurations
7. Check for leaked resources
8. Use CDN
9. Keep clusters in different zones for high availability
10. Node that does encryption and decryption should have more resources than other nodes
11. Gateway timeouts, service timeouts

Security Considerations:

1. Always use latest components - vulnerability remediations
2. JWT tokens - Change secrets often - can use asymmetric with LetsEncrypt certificates, Keep blacklisted tokens for handling logouts, Use refresh tokens for user's seamless experience
3. SQL Injections should not be allowed in code
4. Loggers should never have sensitive info
5. Code should never have sensitive info
6. Strong CORS policy, Same origin cookies
7. Password suggestions for Sentences
8. Check firewalls, open ports - whitelist IPs wherever required
9. DDOS settings
10. Use strong cryptographic algorithms like Bcrypt

Proactive Monitoring:

1. Monitor occurrences of Full GC
2. Check Heap Dump & Thread Dump whenever required
3. Monitor health of services using Grafana, Monitor using Dynatrace/Geneos
4. Alerts

5. Monitor Payment failures - Reconciliation with vendor, Record request response of vendors
6. Record payment failure in Error Audit

KPIs:

1. Tickets booked in a day
2. Tickets booked for movies for different partners - Publish this result to Partners - This will enable them to add more screens as per the demand
3. Tickets booked with each offer