Pre Requisites

1. AWS account

2. AWS CLI installed in the system

3. Docker installed in the system

4. Maven setup in the system

5. Nodejs installed in the system

--------------Run application in local--------------

Step 1: Database setup

1. docker run --name tmsdb -e MYSQL\_DATABASE=tmsdb -e MYSQL\_ROOT\_PASSWORD=root -e MYSQL\_ROOT\_HOST=% -p 3306:3306 mysql/mysql-server:5.7

Step 2: Extract files

1. Extract 539617\_practise\_case\_study

Step 3: kafka setup

1. Open command prompt window

1. Navigate to travel\_management\_system-microservices\tms\_kafka\_service

2. Run 'docker-compose up'

Step 4: local dynamodb setup

1. Open command prompt window and run 'docker run -p 8000:8000 --name dynamolocal amazon/dynamodb-local -jar DynamoDBLocal.jar -inMemory -sharedDb'

2. Open browser and go to http://localhost:8000/shell/# to open local dynamodb console.

3. Run the script in the editor from '539617\_practise\_case\_study\scripts\dynamodb\_script'

Step 5: Build and run microservices

1. Open command prompt window

2. Navigate to \539617\_practise\_case\_study\travel\_management\_system-microservices\tms\_eureka\_service

3. Run 'mvn clean package'

4. Navigate to \target folder and run 'java -jar tms\_eureka\_service-0.0.1-SNAPSHOT.jar'

5. Repeat step 3 and 4 for below services as well in the same order with the corresponding jar files

tms\_config\_service

tms\_zuul\_service

tms\_auth\_service

tms\_ride\_service

tms\_driver\_service

tms\_rider\_service

6. Navigate to 539617\_practise\_case\_study\travel\_management\_system-ui

7. Run 'npm install'

8. ng serve --o

--------------Run application in AWS--------------

Step 1: VPC, subnets and security group setup

1. Login to AWS Console and navigate to VPC service

2. Create a VPC

3. Create 4 subnets for the created VPC (2 for public and 2 for private)

4. Create a security group and assign the VPC created in step 1. This will be public security group

5. Go to Inbound Rules tab and add below ports for public access outside VPC

1111

2222

3333

4444

8761

8888

8080

9092

80

6. Create another security group and assign the VPC created in step 1. This will be private security group

7. Go to Inbound Rules tab. Select 'MYSQL/Aurora' in Type and 'Custom' in Source and select public security group created in step 2

Step 2: RDS setup

1. create a MySql database as below

VPC: Created in previous step

VPC Security group: private security group created in previous step

2. Note down the database, username and password

Step 3: DynamoDB setup

1. Navigate to DynamoDB service and create a DynamoDB table as below

Name: ridelog

KeySchema

AttributeName: rideid; KeyType: HASH; AttributeType:S

AttributeName: ridedate; KeyType: RANGE; AttributeType:S

GlobalSecondaryIndexes

IndexName: riderid; AttributeName: riderid; KeyType: HASH; ProjectionType: ALL

IndexName: driverid; AttributeName: driverid; KeyType: HASH; ProjectionType: ALL

IndexName: riderstatus; AttributeName: ridestatus; KeyType: HASH; ProjectionType: ALL

Step 4: Kafka setup

1. Please follow the steps from https://medium.com/@maftabali2k13/setting-up-a-kafka-cluster-on-ec2-1b37144cb4e

Step 3: Route53 setup for Eureka server

1. Navigate to EC2 service and allocate an elastic ip

2. Navigate to Route53 service and go to 'Hosted Zones'

3. Create a hosted zone 'tms.net'

4. Create a record set as below

name: txt.ap-south-1

type: TXT - text

value: "ap-south-1a.tms.net"

5. Create another record set as below

name: txt.ap-south-1a

type: TXT - text

value: "ec2-15-206-3-195.ap-south-1.compute-1.amazonaws.com"

Step 4: Extract files

1. Extract 539617\_practise\_case\_study zip file

2. Extract travel\_management\_system in microservices folder

3. Extract fse2-assignment1-ui

Step 5: Create repositories in ECR in AWS Console

1. Create below repositories in ECR

eureka

ride

auth

config

driver

rider

zuul

tmsui

Step 6: Login to AWS ECR from command prompt

1. Open command prompt window and run 'aws ecr get-login --region ap-south-1'

2. Copy and run the returned 'docker login' command

3. If it errors, remove '-e none' from 'docker login' command and try again

Step 7: Build and push docker images to ECR from local system

1. Open command prompt window

2. Navigate to \539617\_practise\_case\_study\travel\_management\_system-microservices\travel\_management\_system\tms\_eureka\_service

3. Run 'mvn clean package'

4. Run 'docker build -t eureka .'

5. Run 'docker tag eureka:latest 296301478287.dkr.ecr.ap-south-1.amazonaws.com/eureka:latest'

6. Run 'docker push 296301478287.dkr.ecr.ap-south-1.amazonaws.com/eureka:latest'

7. Repeat step 1 to 6 for below services. Note: Replace 'eureka' with appropriate service name

tms\_ride\_service

tms\_auth\_service

tms\_config\_service

tms\_driver\_service

tms\_rider\_service

tms\_zuul\_service

8. Navigate to \539617\_practise\_case\_study\fse2-assignment1-ui\fse2-assignment1-ui

9. Run 'docker build -t tmsui .'

10. Run 'docker tag tmsui:latest 296301478287.dkr.ecr.ap-south-1.amazonaws.com/tmsui:latest'

11. Run 'docker push 296301478287.dkr.ecr.ap-south-1.amazonaws.com/tmsui:latest'

Step 8: Create Cluster in ECS

1. Navigate to ECS

2. Create a cluster for eureka service as below

Template: EC2 Instance + Networking

Name: eureka

EC2 instance type: t2.large

Number of instances: 2

VPC: created vpc

Subnets: Both created public subnets

Security group: created public security group

3. Repeat step 2 for the below services

eureka

config

zuul

auth

ride

driver

rider

tmsui

Step 9: Create tasks in ECS

1. Create a task for eureka as below

Name: eurekatask

Container Definition:

Name: eureka

Image: give the image url for eureka from ECR

Port Mappings:

Host Port: 80

Container Port: 8761

2. Create a task for config as below

Name: configtask

Container Definition:

Name: config

Image: give the image url for config from ECR

Port Mappings:

Host Port: 8888

Container Port: 8888

Environment variables

EUREKA\_SERVER: http://<ELASTIC IP CREATED>/eureka

3. Create a task for zuul as below

Name: zuultask

Container Definition:

Name: zuul

Image: give the image url for zuul from ECR

Port Mappings:

Host Port: 80

Container Port: 8080

Environment variables

EUREKA\_SERVER: http://<ELASTIC IP CREATED>/eureka

4. Create a task for auth as below

Name: authtask

Container Definition:

Name: auth

Image: give the image url for auth from ECR

Port Mappings:

Host Port: 1111

Container Port: 1111

Environment variables

EUREKA\_SERVER: http://<ELASTIC IP CREATED>/eureka

DATASOURCEURL: jdbc:mysql://<DATABASE ENDPOINT>:3306/<DATABASE NAME>?useSSL=false

DBUSER: <DB USER>

DBPASSWORD: <DATABASE PASSWORD>

EUREKA\_SERVER: http://<ELASTIC IP CREATED>/eureka

5. Create a task for ride as below

Name: ridetask

Container Definition:

Name: ride

Image: give the image url for ride from ECR

Port Mappings:

Host Port: 2222

Container Port: 2222

Environment variables

EUREKA\_SERVER: http://<ELASTIC IP CREATED>/eureka

DATASOURCEURL: jdbc:mysql://<DATABASE ENDPOINT>:3306/<DATABASE NAME>?useSSL=false

DBUSER: <DB USER>

DBPASSWORD: <DATABASE PASSWORD>

EUREKA\_SERVER: http://<ELASTIC IP CREATED>/eureka

6. Create a task for driver as below

Name: drivertask

Container Definition:

Name: driver

Image: give the image url for driver from ECR

Port Mappings:

Host Port: 3333

Container Port: 3333

Environment variables

EUREKA\_SERVER: http://<ELASTIC IP CREATED>/eureka

DATASOURCEURL: jdbc:mysql://<DATABASE ENDPOINT>:3306/<DATABASE NAME>?useSSL=false

DBUSER: <DB USER>

DBPASSWORD: <DATABASE PASSWORD>

EUREKA\_SERVER: http://<ELASTIC IP CREATED>/eureka

7. Create a task for rider as below

Name: ridertask

Container Definition:

Name: rider

Image: give the image url for rider from ECR

Port Mappings:

Host Port: 4444

Container Port: 4444

Environment variables

EUREKA\_SERVER: http://<ELASTIC IP CREATED>/eureka

DATASOURCEURL: jdbc:mysql://<DATABASE ENDPOINT>:3306/<DATABASE NAME>?useSSL=false

DBUSER: <DB USER>

DBPASSWORD: <DATABASE PASSWORD>

EUREKA\_SERVER: http://<ELASTIC IP CREATED>/eureka

8. Create a task for tmsui as below

Name: tmsuitask

Container Definition:

Name: tmsui

Image: give the image url for tmsui from ECR

Port Mappings:

Host Port: 80

Container Port: 80

Step 10: Create services

1. In ECS, create services for each of the tasks in the same order

eureka

config

zuul

auth

ride

driver

rider

tmsui

Once all the services are up and running, the application should be accessible from the public endpoint of tmsui.