**E-STOCKMARKET APPLICATION**

**E-StockMarket** Application is a restful Microservice application, where it allows users to manage the stocks like create, view stock price details and company details.

**Backend Architecture Flow**



**Table Structure**



**Zuul Service**

Create Gateway-Service Zuul Application, register it in Eureka server, validate token, and centralize the authentication incoming request from client service.

**Eureka Server**

Holds the information about all client service applications.  
  
  
  
Segregation of READ & WRITE [CQRS PATTERN]

**Command Service**

Create microservices that responsible to handle write operations (User, Company, Sector, Stocks).

**Query Service**

Create microservices that responsible to handle read operations (User, Company, Sector, Stocks).

**EVENT SOURCING**

**Kafka Topology**Topic created for each services handling events in Kafka cluster. That would be Kafka Producer. Whenever the Services data get updated in Write DB the events get updated, it will add to the topic and consuming microservices will subscribe to it and store in Read DB.

**Elasticsearch-Logstash-Kibana**

Collect and centralized logging in three open-source projects for analysis in various environments, search indexes, visualize data with charts and graphs.  
  
**Prometheus**

All application metrics data are accessible via an actuator endpoint named /prometheus. The Prometheus server also pull this endpoint on a regular basis to obtain metrics data.

The prometheus endpoint will begin to appear on the actuator endpoint-discovery page (<http://localhost:6092/actuator>).

**Deployment Strategy with Docker**

Created Docker Networking to allow multiple containers to communicate with each other.

docker network create Estockmarket-hackfse  
  
Build image for each microservices

**1. Eureka Service**

mvn clean install package -DskipTests

docker image build -t eureka --rm=true

**2. Zuul Service**

mvn clean install package -DskipTests

docker image build -t zuul --rm=true

**3. Command Service**

mvn clean install package -DskipTests

docker image build -t command-service --rm=true

**4. Query Service**

mvn clean install package -DskipTests

docker image build -t query-service --rm=true

Create and run container using Docker Compose

docker-compose up --build

**Using the API**The API gateway is listening at localhost:6091. You will have to interact with the API gateway, which takes care of the proper routing to one instance of the command or the query side of the application.

**Overview  
  
Query MicroService**

|  |  |  |
| --- | --- | --- |
| **API Endpoint** | **Method** | **Description** |
| /api/v1.0/query/user/authenticate | GET | Authenticate the user |
| /api/v1.0/query/user/generateOtp | GET | Generate the otp |
| /api/v1.0/query/user/validateOtp | GET | Validate the otp |
| /api/v1.0/query/market/company/getall | GET | List all companies that are currently managed |
| /api/v1.0/query/market/company/info/{code} | GET | Fetch company details through company code |
| /api/v1.0/query/market/company/view/{code} | GET | View company latest stock price through company code |
| /api/v1.0/query/market/sector/getAll | GET | List all sector that are currently managed |
| /api/v1.0/query/market/stock/aggregate | GET | Get min, max and avg stock price |
| /api/v1.0/query/market/stock/get | GET | Fetch company stocks based on time frame |

**Command MicroService**

|  |  |  |
| --- | --- | --- |
| **API Endpoint** | **Method** | **Description** |
| /api/v1.0/command/user/register | POST | Register a new user |
| /api/v1.0/command/market/company/register | POST | Register new company |
| /api/v1.0/command/market/company/delete/{code} | DELETE | Delete the company details through company code |
| /api/v1.0/command/market/sector/create | POST | Create a sector |
| /api/v1.0/command/market/sector/{id} | DELETE | Delete sector details by sector id |
| /api/v1.0/command/market/stock/add | POST | Create stocks |
| /api/v1.0/command/market/stock/update | POST | Update stocks |
| /api/v1.0/command/market/stock/{id} | DELETE | Delete stock details based on stock id |
|  |  |  |