Alert Monitoring System

Implementation Note

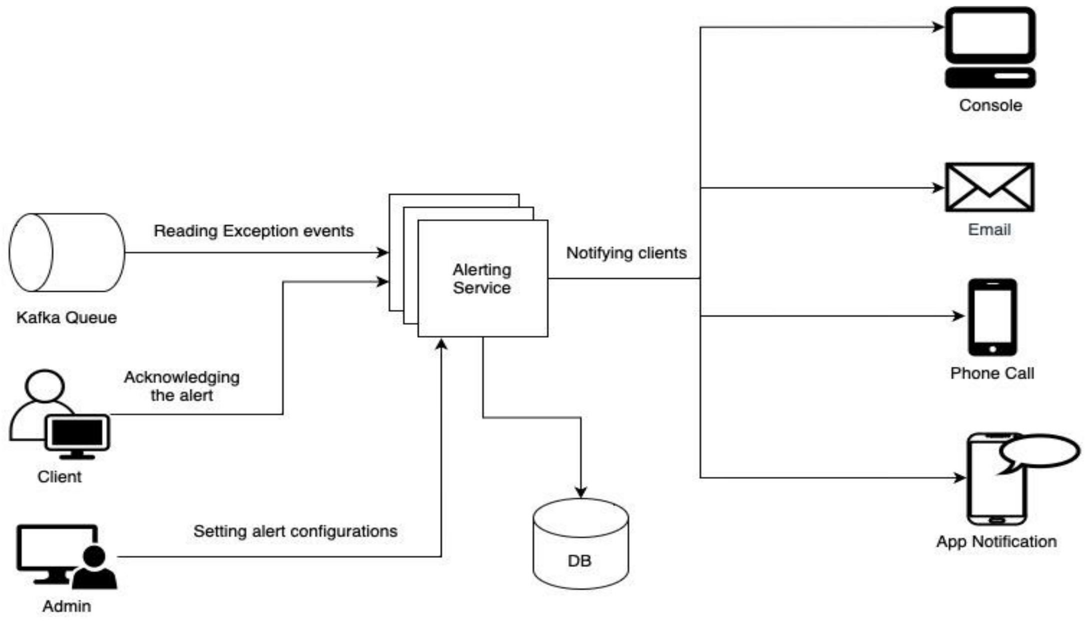
Problem Statement

Consider a microservice env where several systems use 1 central system for all their alert use cases: Design and implement a system which can capture events generated by any system or user triggered (for simplicity), and raise an alert according to alert configuration.

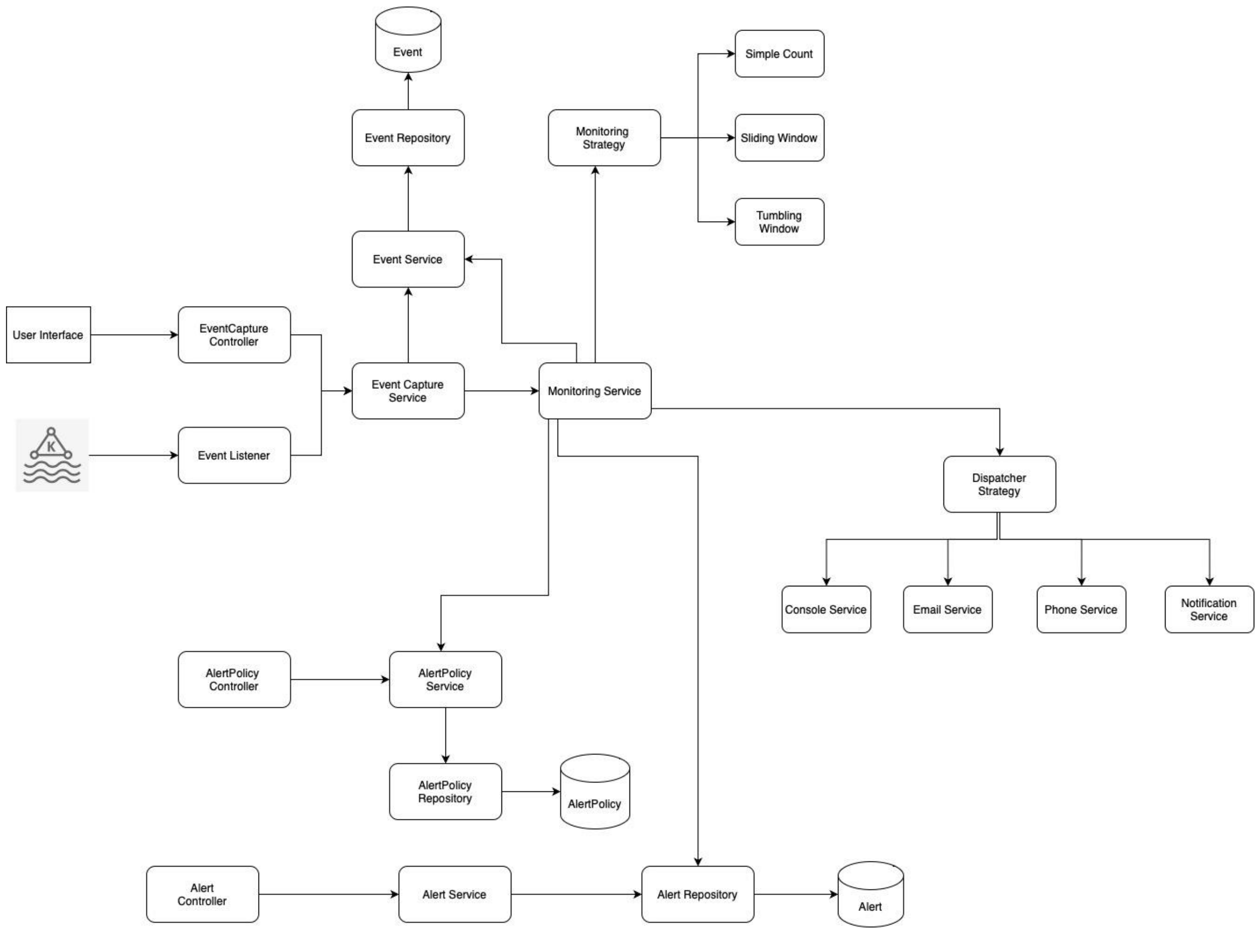
Types of Alert Configuration

- SIMPLE_COUNT (which consists of count)
- TUMBLING_WINDOW (which consists of count and windowSizeInSecs, Eg: 10 events in 1 hour bucket, you can assume that the bucket starts at 00:00 hour of the day)
- SLIDING_WINDOW (which consists of count and windowSizeInSecs)

High Level Diagram



Low Level Diagram



Data Models

Alert Config

Name	Data Type	Description
clientId	String	clientId of the client service for which alert is configured
alertType	ENUM (SIMPLE_COUNT, TUMBLING_WINDOW, SLIDING_WINDOW,…)	type of the alert
eventType	ENUM (PAYMENT_EXCEPTION,	type of the exception thrown by client service, for which alert is configured

	USERSERVICE_EXCEPTION, ...)	
createdAt	long	created time of this alert configuration.
count	long	number of exceptions to consider for triggering an alert.
windowSize	long	Size of the window to consider while triggering an alert
communicationInfo	List <CommunicationInfo>	List of communication channels with messages

CommunicationInfo

Name	Data Type	Description
mode	ENUM (PHONE, EMAIL, CONSOLE)	mode of the communication
message	String	message that needs to be communicated

Event

Name	Data Type	Description
evenType	ENUM (PAYMENT_EXCEPTION, USERSERVICE_EXCEPTION,...)	type of the exception thrown by the client service
clientId	String	clientId of the clientService which has thrown the exception event
timeStamp	long	timeStamp of the exception

Alert

Name	Data Type	Description
evenType	ENUM (PAYMENT_EXCEPTION, USERSERVICE_EXCEPTION,.	type of the exception thrown by the client service

	..)	
clientId	String	clientId of the clientService which has thrown the exception event
startTime	long	timeStamp of the first event of the alert
endTime	long	timeStamp of the last event of the alert
status	ENUM (TRIGGERED , ACKNOWLEDGED, IGNORED, RESOLVED)	status of the alert

API Design

Create Alert Config API

Description : To set alert configuration.
Endpoint : /ams/configs
Type : POST

Request

```
{
        client: X,
        eventType: PAYMENT_EXCEPTION,
        alertConfig {
                type: TUMBLING_WINDOW,
                count: 10
                windowSize: 10
        },
        “dispatchStrategyList”: [ {
                        “mode”: CONSOLE,
                        “message” : “issue in payment”
                },{
                        “mode”: EMAIL,
                        “message” : “payment exception threshold breached”
                }
        ]
}
```

Response

```
{
   “status” : “success”,
   “message” : “Alert config got successfully created with id : 90df8f0cd6bfb275”
}
```

Update Alert Config by Config ID API

Description : To update alert configuration by configId.
Endpoint : /ams/configs/{configId}
Type : POST

Request

```
{
        client: X,
        eventType: PAYMENT_EXCEPTION,
        alertConfig {
                type: TUMBLING_WINDOW,
                count: 10
                windowSizeInSecs: 10
        },
        “dispatchStrategyList”: [ {
                        “type”: CONSOLE,
                        “message” : “issue in payment”
                },{
                        “type”: EMAIL,
                        “message” : “payment exception threshold breached”
                }
        ]
}
```

Response

```
{
   “status” : “success”,
   “message” : “Alert configuration with id : 90df8f0cd6bfb275, got updated successfully”
}
```

Create Exception API

Description : To send an exception event.
Endpoint : /ams/events
Type : POST

Request

```
{
        client: X,
        eventType: PAYMENT_EXCEPTION,
        timeStamp  : 1694859869000
}
```

Response

```
{
   “status” : “success”,
   “message” : “Recorded PAYMENT_EXCEPTION event for client X”
}
```

Acknowledge Alert API

Description : To acknowledge an alert by alertId event.
Endpoint : /ams/alerts/{alertId}?status=RESOLVED
Type : GET

Request Parameter

status: RESOLVED

Response

```
{
   “status” : “success”,
   “message” : “Alert with  id : b0df8f0cd6bfb275 got successfully acknowledged”
}
```

Tech stack used

Framework : Spring Boot ; Version : 3.0.7
Language : Java ; Version: 17
Database : MongoDb